

目 录

基 本 篇

第一章 引言	3
§1.1 T _E X 和 L ^A T _E X	3
§1.2 天元软件	5
§1.3 CJK 宏包	6
§1.4 排版过程	6
§1.5 T _E X 中的长度	7
1.5.1 固定长度	7
1.5.2 弹性长度	7
第二章 准备文稿	9
§2.1 基本格式	9
§2.2 输入特殊字符	11
2.2.1 几个特殊字符	11
2.2.2 空格与换行	12
2.2.3 引号、连字号、破折号	13
2.2.4 连体字	14
2.2.5 句号后的空白	14
2.2.6 非英文字母及重音符号	14
§2.3 分组与环境	15
§2.4 分段	16
§2.5 分行和分页	16
2.5.1 分行	16
2.5.2 分页	17
§2.6 水平间距、竖直间距	18

2.6.1 水平间距	18
2.6.2 竖直间距	20
§2.7 与段落有关的距离	21
2.7.1 首行缩进	21
2.7.2 段落间距	22
2.7.3 伸展行距	22
第三章 文字模式	23
§3.1 中文字体	23
§3.2 西文字体	24
3.2.1 字体属性	24
3.2.2 选择字体尺寸	26
§3.3 居中	28
§3.4 参考文献	28
§3.5 制表位	30
§3.6 表格	32
3.6.1 表格环境	32
3.6.2 样例	34
§3.7 脚注	36
第四章 数学公式	37
§4.1 概述	37
§4.2 行内公式	39
§4.3 行间公式	40
§4.4 上标和下标	41
§4.5 分式	43
§4.6 根式	45
§4.7 求和、积分	46
§4.8 数学重音符号	47
§4.9 上划线、下划线及类似符号	48
§4.10 堆叠符号	49
§4.11 可以变大的定界符	51
§4.12 矩阵	53
§4.13 单行公式与多行公式	55
§4.14 数学字体	59

4.14.1 选择数学字体	59
4.14.2 希腊字母	60
4.14.3 数学黑体	60
§4.15 数学符号表	61
4.15.1 二元运算符	62
4.15.2 关系运算符	62
4.15.3 箭头符号	63
4.15.4 其他符号	64
4.15.5 具有两种尺寸的符号	64
4.15.6 函数名	64
第五章 常用的文档的类别与版式	66
§5.1 文档类别命令中的可选项	66
5.1.1 指定基本字体尺寸	66
5.1.2 指定纸张大小	67
5.1.3 其它一些选项	67
§5.2 章节	68
§5.3 文章标题	70
§5.4 摘要	72
第六章 自定义与改错	75
§6.1 自定义命令	75
§6.2 给计数器和长度赋值	77
§6.3 出错信息	78
§6.4 警告信息	80
提 高 篇	
第七章 图形	85
§7.1 图形与坐标系	85
§7.2 基本绘图命令	86
7.2.1 直线和矢量线	86
7.2.2 圆和圆角矩形	88
7.2.3 图形中的盒子	90

7.2.4 图形中的文本	92
7.2.5 曲线	92
§7.3 子图	93
§7.4 天元的绘图功能	98
§7.5 插入外部图形	104
7.5.1 使用 emTeX 时插入图形	104
7.5.2 使用 PCTeX32 时插入图形	105
7.5.3 图形包	107
7.5.4 插入图形的基本命令	109
7.5.5 放大和缩小	110
7.5.6 水平翻转和旋转	111
§7.6 浮动表格和图形	114
§7.7 其他绘图软件包介绍	115
第八章 如何使文本竖向对齐	119
§8.1 使文本居左或居右	119
§8.2 引文	120
§8.3 抄录	121
§8.4 罗列	121
8.4.1 三种罗列环境	121
8.4.2 改变默认的罗列条目标签	123
§8.5 广义罗列环境	125
8.5.1 标准标签	125
8.5.2 广义罗列环境的样式参数	126
8.5.3 平凡罗列环境	128
§8.6 盒子	129
8.6.1 LR 盒子	129
8.6.2 LR 盒子的升降	131
8.6.3 标尺盒子	132
8.6.4 子段盒子与小页环境	133
§8.7 制表位的高级技巧	134
§8.8 表格的高级技巧	135
8.8.1 更多的表格参数	135
8.8.2 几个表格样例	137

§8.9 脚注与边注	140
8.9.1 自动编号的脚注	140
8.9.2 指定编号的脚注	141
8.9.3 禁止模式中的脚注	141
8.9.4 边注	142

第九章 数学公式排版的一些技巧 143

§9.1 巧妙使用阵列环境	143
§9.2 单行公式与多行公式	146
§9.3 数学模式中的参数	148
§9.4 数学模式中的字体尺寸	149
§9.5 数学排版的国际标准	150
§9.6 定理定义的排版	151
§9.7 amsmath 宏包简介	153
§9.8 公式中的文本 (amsmath)	155
§9.9 单个公式 (amsmath)	156
§9.10 方程组 (amsmath)	158
9.10.1 gather 环境	158
9.10.2 align 环境	159
9.10.3 flalign 环境	160
9.10.4 alignat 环境	160
9.10.5 gathered, aligned 和 alignedat 环境	161
9.10.6 cases 环境	162
§9.11 矩阵 (amsmath)	163
§9.12 多重数学符号 (amsmath)	164
9.12.1 多重角标	164
9.12.2 多重积分	165
9.12.3 叠置重音符号	166
9.12.4 省略号	167
§9.13 分式 (amsmath)	168
9.13.1 普通分式	168
9.13.2 连分式	168
9.13.3 二项式系数	169
9.13.4 自定义分式类命令	169

§9.14 函数(算子)名 (amsmath)	170
9.14.1 已定义的函数名	170
9.14.2 定义新的函数名	171
§9.15 其他功能 (amsmath)	171
9.15.1 公式中的空白间隔	171
9.15.2 调整根式指数的位置	172
9.15.3 调整公式编号的竖直位置	172
9.15.4 特殊的上下标(上下限)	173
9.15.5 不可断行的区间符	175
§9.16 交换图	175
9.16.1 用宏包 <code>amscd</code> 画交换图	175
9.16.2 用宏包 <code>diagrams</code> 画交换图	177
第十章 新字体选择方案 (NFSS)	180
§10.1 NFSS 中的字体属性	180
§10.2 简化的字体选择命令	183
§10.3 属性的默认值	185
§10.4 定义新的字体命令	185
§10.5 定义新的数学字体命令	186
10.5.1 数学字母字体	186
10.5.2 数学符号字体	187
§10.6 在 NFSS 下指定字体	189
§10.7 编码命令	192
§10.8 计算机现代字体	194
10.8.1 简介	194
10.8.2 使用 CM 字体	195
第十一章 文档的布局及相互联系	197
§11.1 分栏	197
§11.2 单、双面	198
§11.3 与公式有关的选项	199
§11.4 页版式	200
11.4.1 页面布局	200
11.4.2 指定页眉内容	202
11.4.3 页码	202

§11.5 目录表及图表清单	203
§11.6 文档的分割处理	205
11.6.1 \input 命令	205
11.6.2 \include 命令	205
11.6.3 人机交互命令	207
§11.7 交叉引用	208
11.7.1 交叉引用	208
11.7.2 索引记录	209
 第十二章 CJK 与 CCT	 211
§12.1 CJK	211
§12.2 CCT 系统简介	215
12.2.1 CCT 系统概况	215
12.2.2 CCT 的排版命令	216
12.2.3 Windows 下的 CCT	218
12.2.4 天元与 CCT 的互相转换	218
 第十三章 输出到屏幕、投影仪或互联网	 220
§13.1 如何显示彩色	220
§13.2 如何使用 pdfTeX	221
13.2.1 pdfL ^A T _E X	222
13.2.2 生成适合屏幕阅读的 pdf 文件	226
13.2.3 生成适合投影仪的 pdf 文件	228
§13.3 用 slides 制作投影片	229
§13.4 生成 html 文件	233
 附录一 T _E X 系统的安装	 237
A1.1 自动安装 MiKTeX	237
A1.2 如何安装光盘 T _E X	240
A1.3 MiKTeX 及其相关软件的安装与配置	241
A1.3.1 安装 MiKTeX	241
A1.3.2 安装 WinEdt	247
A1.3.3 WinShell 的安装与配置	247
A1.3.4 天元的安装与运行	248
A1.3.5 关于天元的配置	249

A1.3.6 安装CJK	250
A1.4 如何在DOS环境下安装和使用emTeX	251
A1.5 如何在PCTeX32中使用天元	255
 附录二 \LaTeX 命令简介	 258
 附录三 字体表	 319
 参考文献与网站	 327
 索引	 329

基本篇

第一章 引言

计算机的发展带动了各行各业的发展,使很多行业出现了革命性的变化,例如印刷出版业现已告别铅与火的时代,普遍使用计算机排版系统.

在计算机排版系统出现之前,人们发表文章或出版书籍时是作者将手稿提供给编辑部或出版社,由专职编辑人员在手稿上作文字修改并添加排版指令,交排版工人排出校样,由作者校对后再返回编辑重复上述过程,一般要重复几次,每次重复还有可能出现新的排版错误.对排好的校样,如果要更改版面设置,就需要重排,工作量是很大的.有了计算机排版系统,情况就大不相同了,录入人员(或作者本人)把原稿输入计算机,编辑人员添加排版指令后,可以直接输出用于印刷的胶片.改变字体、版面等设置是很简单的操作.

目前,世界上已经有许多大大小小的排版系统,各有其特点和适用范围,例如方正电子出版系统已是国内大多数报社的首选系统,而普通用户在编排要求不高的稿件时,使用所见即所得的 Word、WPS 等软件也不失为合适的选择.

本书介绍的 TeX 系统是一种使用方便、价格低廉的排版软件,当排版论文、报告和书籍时,其输出质量并不逊色于价格昂贵的大型系统,在某些方面(例如排版数学公式)仍是排版质量最好的系统.

§1.1 TeX 和 LaTeX

TeX 系统是由美国 Stanford 大学教授 Donald E. Knuth 研制的计算机排版软件系统. Knuth 有一个中文名字——高德纳,他在国际上既是著名的数学家又是著名的计算机专家,是享有盛誉的计算机程序设计系列专著 The Art of Computer Programming 的作者.按着他的计划,这套书总共六册.前三册出版后,Knuth 将修订的第二册第二版手稿交出版社排版,但对收到的校样很不满意,因为这时出版社已开始用计算机代替手工排版,当时的字形和版面都很难看,每次的校改也非常麻烦.为了以后出书的方便,他放下手头的工作,开始设计一套高质量的计算机排版软件,花费大量的精力和时间后,研制成功了这套闻名于世的 TeX 系统.稍后他又撰写了一整套 TeX 手册,既讲 TeX 的使用方法,又讲设计原理,这套书也

成了享有盛誉的经典之作. 更难能可贵的是他并没有利用 $\text{T}_{\text{E}}\text{X}$ 系统去发财致富, 而是无私地把源代码向用户公开.

Knuth 为其研制的软件命名为 $\text{T}_{\text{E}}\text{X}$, 取意于希腊词根 $\tau\epsilon\chi$, 因此名称中的 X 应读 χ 的音, 即 $\text{T}_{\text{E}}\text{X}$ 的发音为 [tex] ([x] 的发音类似于汉语拼音的 h) 或 [tek] 而不是 [teks], 这也使得该软件的名称在外形和读音上都不同于另一个软件 $\text{T}_{\text{E}}\text{X}$. 在纯文本环境中, 通常将 $\text{T}_{\text{E}}\text{X}$ 写成 TeX .

利用 $\text{T}_{\text{E}}\text{X}$ 系统编写手稿, 除了正文内容之外, 还需加入一些排版命令. 与大型排版系统不同的是, 这些排版命令通常不是编辑人员加入的, 而是由作者本人完成的.

$\text{T}_{\text{E}}\text{X}$ 提供的排版命令功能强大, 用户可以直接使用这些命令, 也可以发挥创造性, 利用已有的功能自行定义新的命令, 以适合特定的需要.

$\text{T}_{\text{E}}\text{X}$ 系统提供了 300 多条基本命令 (其中有很大部分是键盘上没有的特殊符号的代码), 功能虽然强大, 但使用不够方便. 后来在这些基本命令的基础上, 又定义了 600 多条复合命令, 构成名为 Plain $\text{T}_{\text{E}}\text{X}$ 的宏包 (软件包), 当人们专指 $\text{T}_{\text{E}}\text{X}$ 而不是衍生版本时, 实际指的就是 Plain $\text{T}_{\text{E}}\text{X}$.

Plain $\text{T}_{\text{E}}\text{X}$ 虽然比 “原始系统” 易用, 但排版复杂的版面或公式时仍需书写大量的命令, 还是不够方便, 因此国外许多人利用 $\text{T}_{\text{E}}\text{X}$ 的宏定义功能进行二次开发, 产生了一些 $\text{T}_{\text{E}}\text{X}$ 系统的衍生版本, 其中最著名的是由美国数学会 (AMS) 组织人员编写的 $\text{A}_{\text{M}}\text{S}-\text{T}_{\text{E}}\text{X}$ 和作者为 Leslie Lamport 的 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$. 前者的特点是容易排版复杂的数学公式, 后者适合于排版普通文章及书籍. 但若把两者的优点组合起来则更符合人们愿望, 于是又出现了兼容于 $\text{A}_{\text{M}}\text{S}-\text{T}_{\text{E}}\text{X}$ 而又包含了 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 优点的衍生版本, 但没有广泛流行, 倒是 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 由于在新版本 ($\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$) 中可以加载 $\text{a}_{\text{m}}\text{s}_{\text{m}}\text{a}_{\text{t}}\text{h}$ 宏包, 基本包含了 $\text{A}_{\text{M}}\text{S}-\text{T}_{\text{E}}\text{X}$ 的优点而大为流行, 占据了 $\text{T}_{\text{E}}\text{X}$ 领域的重要位置, 所以本书重点介绍 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$ 系统, 后文中的 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 实际是指 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$.

$\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 系统的特色功能之一是它的自动编号功能. 文章、书籍的章、节、段落以及公式、图表、文献、页码等均可自动编号, 这给作者带来很大方便, 例如增添或删除一个带有编号的公式, 其他的文字不用任何修改, 所有编号都会自动改变, 对编号及其所在页码的引用也都会自动改变. $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 系统还可以自动生成目录页, 可以自动生成索引附录.

为行文方便, 本书用 $\text{T}_{\text{E}}\text{X}$ 泛指 Plain $\text{T}_{\text{E}}\text{X}$ 、 $\text{A}_{\text{M}}\text{S}-\text{T}_{\text{E}}\text{X}$ 和 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$.

$\text{T}_{\text{E}}\text{X}$ 排版系统出现后受到了科学工作者喜爱, 因为它能排出复杂的图表和精美的数学公式, 到目前为止, 国内外公认的数学公式排得最好的排版软件仍是 $\text{T}_{\text{E}}\text{X}$ 系统. 许多国际专业学会的期刊杂志都欢迎作者使用 $\text{T}_{\text{E}}\text{X}$ 系统投稿, 一些出版社也使用 $\text{T}_{\text{E}}\text{X}$ 系统出版书籍.

TeX 的输出文件是 DVI (DeVice Independent, 设备无关) 文件, 这种文件在针式打印机、喷墨打印机、激光打印机以及照排机上的输出版面是完全相同的, 仅仅是文字或图形的分辨率因设备的不同而有所区别.

随着因特网的急速发展, 已有软件可将 TeX 文件转换成 HTML 文件, 放到网页中通过浏览器阅读.

为了用户安装和使用的方便, 开发者将各种 TeX 系统包装起来并添加一些功能, 做成发行版本, 常见的有 MiKTeX、emTeX、PCTeX32、fpTeX、TeXTeX 等版本. 其中 PCTeX32 是商业软件, 其它是免费软件, 可从网上自由下载.

§1.2 天元软件

TeX 是面向西文的排版系统, 不能直接用于中文, 为了充分利用 TeX 系统的功能, 国内的一些学者开发了几种中文预处理系统, 只需用这些预处理系统处理一下含有中文的原稿, 就可以使用 TeX 系统了. 人们通常把中文预处理系统与 TeX 系统合称为中西文 TeX 排版系统. 其中最有名的是 CCT 系统和天元系统, 两者都是免费软件, 都有广泛的人群在使用, 不过因为设计原理不完全相同, 因此它们并不兼容.

天元的最早开发者是肖刚, 以后又有多位数学家参与改进, 目前由陈志杰维护. 为了感谢数学天元基金对我国数学发展的支持, 将此软件取名“天元”. 所以称为天元 (TianYuan) 中西文排版系统, 它是遵循 GNU GPL 的自由软件, 可以从 www.math.ecnu.edu.cn 下载.

对于熟悉 TeX 的人来说, 使用天元不需任何新的学习和训练, 因为它只是增加了一些易懂易记的中文字体字型控制命令而已. 这些控制命令由倒斜线 \ 和紧跟着的一个汉字组成, 根据这个汉字的含义就可明了该控制命令的作用.

天元可以应用于用 GB 码或 GBK 码写成的源文件. GBK 码就是扩展的国标码 (GB13000), 它包含 21 000 多个汉字, 原来的国标码 GB (GB2312) 只包含 6 000 多个汉字, 是 GBK 的一个子集. GBK 码中包括了几乎所有的繁体汉字、异体字、古体字以及日文和韩文中的汉字.

天元内嵌了 Tydraw, 这是由肖刚设计并由陈志杰扩展完善的在 TeX 环境里使用的绘图软件, 其特点是能利用 TeX 画出精确的插图, 除了可以绘制一些常规图形外, 特别是可以画出用参数方程表示的曲线, 这是其它 TeX 绘图软件所没有的功能.

天元可以使用 TrueType 字体或矢量、曲线轮廓字体, 可以自定义汉字大小, 对输出密度没有限制, 既可以在各类普通打印机上输出, 又可以在出版系统高密

度的照排机上出片, 达到专业级的印刷质量.

本书以及作者们撰写的其他中文论文和书籍都是利用天元排版完成的.

§1.3 CJK 宏包

使 L^AT_EX 能处理中、日、韩文字的 CJK 宏包是 Werner Lemberg 开发的, 目前发布的 4.4.0 版已相当成熟, 因此也是一个很好的选择, 本书也将介绍 CJK 宏包的安装与用法, 欢迎读者试用. 不过 CJK 宏包需要 1998 年 6 月 1 日以后的 L^AT_EX 2_ε 版本, 而且对硬盘容量的需求也很大, 欲试用的读者应该有较高的硬件配置, 而且应该安装本书光盘所提供的 MiKTeX 最新版本.

§1.4 排版过程

排版过程主要有以下几个步骤:

1. 首先是编写或修改文稿(源文件), 在源文件中插入排版命令, 具体的编写要求将在后面的章节中详细介绍. 需要提醒的是必须保证源文件是纯文本格式, 即除了作者直接输入的字符以外, 不能含有其他的東西(例如控制字符). 如果你使用 T_EX 的集成环境(如 WinEdt, WinShell, PCTeX32), 那么有内嵌的编辑软件可供使用, 否则, 有很多文本编辑软件, 如 DOS 环境下的 EDIT 和 Windows 环境下的 NOTEPAD (记事本) 都符合条件. 当使用编辑功能很强的软件如 Word、WPS 等时则要小心, 保存文件时必须选择纯文本格式.

当源文件含有中文或 Tydraw 的图形时, 建议文件扩展名用 `ty`, 不含中文时, 建议文件扩展名用 `tex`.

2. 用天元软件处理源文件, 产生扩展名为 `tex` 的文件(由此可知含有中文的源文件不能用扩展名 `tex`). 若源文件中不含中文及 Tydraw 的图形时, 则可省略这一步. 在集成环境里只要点击“天元”按钮就可以了.

3. 用 L^AT_EX 软件处理 `tex` 文件, 产生 `dvi` 文件. 如果这一步出现错误, 则表明源文件中的某个或某些排版命令有错误, 应返回第一步进行修改.

4. 在屏幕上显示或在打印设备上输出 `dvi` 文件, 察看文件内容或排印的版面是否合乎要求, 如果有不满意之处, 返回第一步进行修改.

5. 上述几步通过后, 就可以正式打印输出或照排制版了.

在 Windows 操作系统下有好几种集成环境可以使用, 在 DOS 环境则可用命令行方式分步完成, 但较麻烦, 使用集成环境则简单得多. 附录一将重点介绍 MiKTeX 2.1 版以及集成环境 WinEdt 或 WinShell 的安装方法. 对于机器配置较低

的读者,则可以安装DOS环境下16位的emTeX,为方便用户,已将常用命令放在批处理程序文件中,模拟成一个集成环境.附录一还介绍了使PCTeX32与天元配合的方案,需要的文件都储存在光盘上.不过后两种解决方案由于版本的限制,不能保证本书的内容都能正确实现.本书还提供了光盘版的TeX,只要在硬盘上安装少量文件,就能在光盘上运行TeX(当然速度较慢),而且可以干净卸载,特别适用于临时借用别人机器运行TeX的用户.

§1.5 TeX 中的长度

TeX 中的长度可分成两类,一类是固定长度,一类是根据排版情况可以伸缩的弹性可变长度.

1.5.1 固定长度

TeX 中固定的长度、宽度或距离用十进制小数和一个长度单位组成,数字可以带有正负号,小数点可采用英美方式(使用圆点句号),也可采用欧洲方式(用逗号作小数点).常用的单位有如下几种:

mm	毫米
cm	厘米, $1\text{ cm} = 10\text{ mm}$
in	英寸, $1\text{ in} = 2.54\text{ cm}$
pt	点(磅), $1\text{ in} = 72.27\text{ pt}$
em	与当前字号有关,相当于大写字母M的宽度
ex	与当前字号有关,相当于小写字母x的高度

当长度为0时,不能只写数字0,必须附上单位,例如写成0mm或0pt等.也可以使用下列长度单位:

bp	大点, $1\text{ in} = 72\text{ bp}$
pc	pica, $1\text{ pc} = 12\text{ pt}$
dd	didot, $1157\text{ dd} = 1238\text{ pt}$
cc	cicero, $1\text{ cc} = 12\text{ dd}$

1.5.2 弹性长度

所谓弹性长度就是根据排版需要可以自动伸长或缩短的长度,这种长度实际上由3个非负的长度组成,一是正常长度.即没有伸缩时的长度,二是伸长时最多可以增加的长度,三是缩短时最多可以减少的长度.定义弹性长度的语法是:

正常值 plus 伸展值 minus 收缩值

为了能自动排版出优美的版面, 在 TeX 中使用了大量的弹性长度. 此外还有一个特殊的弹性长度 `\fill`, 它的正常长度是零, 但可以伸展到任何长度.

第二章 准备文稿

§2.1 基本格式

文稿(即用于排版的源文件)包含两部分内容:一部分是正文,也就是需要排版输出的内容;另一部分是排版控制命令,用于控制版面式样、字体字形等格式.控制命令是用倒斜线引导的字符串.

中文排版命令由倒斜线和一个汉字组成,例如\黑,它使得该命令后面与该命令位于同一分组的汉字都输出为黑体.所有中文排版命令都不带参数.

西文排版命令分为两种,一种是“控制字”,一种是“控制符”.控制字由倒斜线和一个或多个英文字母组成,区分大小写,可以用任何非英文字母的字符(例如空格、数字、括号、标点符号等)表示控制字的结束.控制符由倒斜线和一个符号(非英文字母)组成.有一些西文排版命令带有参数,不可省略的参数放在花括号中,可以省略的参数(可选参数)放在方括号中,这种带参数的命令格式为

`\命令名[可选参数]{不可省略的参数}`

当同一个括号中含有多个参数时,需用逗号分隔参数.为了区分带参数和不带参数的命令,有时把某些(并非全部)不带参数的命令称为**声明**.

源文件的基本格式如下例所示:

```
1 \def\ChineseScale{1000}
2 \input tyinput
3 \documentclass{article}
4 \begin{document}
5 祝贺你, MiKTeX 和天元安装成功了!
6 \end{document}
```

为了叙述方便,在每行行首加了编号,实际输入时绝不能带编号.当源文件不含汉字时,建议用 `tex` 做文件的扩展名,当源文件含有汉字时,建议用 `ty` 做文件的

扩展名, 扩展名与主名之间用“.”分隔. 例如将上面的例子命名为2-1-1.ty (本书例子第一个数字表示章, 第二个数字表示节, 第三个数字表示该例在本节中的序号).

语句

```
\def\ChineseScale{1000}
```

用于控制中文字的大小, 括号中的数字除以 1000 后的值表示对基本字体尺寸的放大倍数. 语句

```
\input tyinput
```

用于读入文件tyinput.tex, 这两行总是形影不离的. 如果正文中不含中文, 可以删除这两行, 或将这两行注释掉, 即在这两行行首添加字符%, 使得这两行对排版不起作用.

第3行、第4行和第6行是必不可少的. 语句

```
\documentclass{article}
```

用于决定文章的版式类别, 花括号中的 article 表示这是一篇普通文章, 若将它改为 book, 则表示排成书籍样式. 不同的版式类别具有不同的默认样式.

第5行处写正文, 可以写多行. 第4行的\begin{document}表示正文的开始, 而第6行的\end{document}表示正文的结束. 这个例子的输出结果是:

祝贺你, MiKTeX 和天元安装成功了!

如果想把上述例中的“简单”两字改为黑体, 只需用花括号将这两字括起来组成一个“分组”(或称集团), 并对它们使用中文黑体命令, 见下例(2-1-2.ty).

```
1 \def\ChineseScale{1000}
2 \input tyinput
3 \documentclass{article}
4 \begin{document}
5 祝贺你, MiKTeX 和{\黑天元}安装成功了!% 注意花括号不要忘记
6 \end{document}
```

输出结果是:

祝贺你, MiKTeX 和**天元**安装成功了!

以 % 开始的语句是注释内容, 注释内容从 % 开始, 到所在行行尾结束, 注释内容对排版没有影响, 也不会产生输出. % 号除表示注释外, 还有一个重要功能, 就是当文稿分行输入时, 如果行尾的空格或回车影响了排版时, 可在该行行尾加上 % 号. 虽然大多数情况下文稿行尾不必加 %, 但若在命令的两个参数间分了行而又要避免行尾隐含的空格时, 就必须在行尾加 % 了.

`\begin{document}` 之前的部分称为导言部分, 之后部分称为正文主体部分. 导言部分的命令是全局性的, 对整个文档起作用.

TeX 源文件是一种自由格式文件, 输入源文件时不必考虑每行的长短, 也不必考虑单词之间空白的多少, 只要符合下面的一些输入要求, TeX 系统就会自动按着你的命令排版. 因此建议在输入源文件时, 为了便于校对和改错, 每行不要太长, 最好在标点符号后面换行, 也不要忘了在西文标点后面应空一格. 各种排版环境的开始命令和结束命令最好是单独占一行, 就象前面的例子一样.

用户如果不想使用 TeX 系统自动确定的页芯大小(参见第 67 页), 可以直接指定页芯大小, 只需在导言区写上命令

```
\setlength{\textwidth}{页芯宽度}
\setlength{\textheight}{页芯高度}
```

其中的宽度和高度可以使用 mm, cm 或 in 作长度单位. 注意页芯宽度不含边注, 页芯高度不含页眉和脚注.

§2.2 输入特殊字符

2.2.1 几个特殊字符

大部分键盘字符都可直接输入, 但字符 “# \$ % { } ~ _ ^ \ | < >” 在 TeX 中有特殊用途, 如果需要排版输出前面 9 个字符, 应按下表对应输入:

输出字符	#	\$	%	{	}	~	_	^	\
输入序列	<code>\#</code>	<code>\\$</code>	<code>\%</code>	<code>\{</code>	<code>\}</code>	<code>\~{}</code>	<code>_{}</code>	<code>\^{}</code>	<code>\backslash</code>

接下去的 3 个字符 “| < >” 如果直接输入, 其打印结果成了 “— i j”, 完全变了样. 为得到正确的输出, 应将它们用美元号 \$ 括起来, 即输入 “`\$ \$< \$ \$> \$`”, 得到的输出为 “| < >”. 这是因为按 TeX 规定, 用美元号 \$ 括起来的部分处于数学模式

(后面会详细介绍), 这 3 个字符只有在数学模式下才能打印.

此外, Plain TeX 中的特殊字符 “@” 在 LaTeX 中变成了普通字符, 可直接输入.

字符 “*” 是一个普通字符, 这是直接输入后排印显示的结果, 如果想要它显示在上下居中的位置如 “ $*$ ”, 可用数学模式输入成 $*$$.

还有一些有用的符号可参见下表.

输出符号	§	¶	†	‡	©	£
输入序列	\S	\P	\dag	\ddag	\copyright	\pounds

为了得到 TeX、LaTeX 与 LaTeX 2_ε 的标志, 可分别输入 “\TeX{}”、“\LaTeX{}” 以及 “\LaTeXe{}”.

下面是使用特殊符号的例子: 为得到 “\$3.50”, 应该输入 “\\$3.50”; 为得到 “50%”, 应该输入 “50\%”.

2.2.2 空格与换行

我们知道, 西文单词是用空格隔开的, 句末的标点之后也应该留有空格. 但应注意, TeX 源文件中的连续多个空格在编译排版时被看作一个空格, 而且与许多“所见即所得”的排版软件(例如 Word, WPS 等)不同, 单个回车(注意, 连续两个回车将被看成段落的结束, 参见 §2.4)仅相当于一个空格, 与最终得到的版面没有关系. 因此你可以放心换行, 不必担心最终结果, TeX 会自动使得左右两端对齐的. 此外, 西文以后的空格标志着单词或句子的结束, 排版时会留有适当的间距, 而汉字中间的空格则在排版时不起作用. 因此换行最好选在句末标点以后, 或在两个汉字之间, 尽量避免选在汉字与西文的交界处. 控制字后面的空格被认为是命令序列的结束标志, 排版输出时不会出现空格所对应的间隔, 如果在控制字后面紧接着放上 _, 即一个倒斜线和一个空格, 则它既表示命令序列的结束, 又会输出一个空格对应的间隔. 控制符后面不需要任何结束标志, 因此在控制符后面的空格会输出相应的空白间隔.

实际的空格是看不见的, 人们在板书或文章中常常用符号 “_” 表示空格, 本书也是如此.

下面的例 2-2-1.ty 让我们比较一下输入空格的效果. 我们输入以下几段文字:

TeX是美国Stanford大学教授Donald E. Knuth研
制的。

TeX 是美国 Stanford 大学 教授
Donald E. Knuth 研制的。

TeX 是 \ 美国 \ Stanford \ 大学 教授
\ Donald \ E. \ Knuth \ 研制的。

输出结果是

TeX是美国Stanford大学教授Donald E. Knuth研制的。

TeX 是美国 Stanford 大学教授 Donald E. Knuth 研制的。

TeX 是美国Stanford 大学教授Donald E. Knuth 研制的。

仔细观察这个例子, 注意汉字之间的空格完全不起作用, 第二句的 TeX 以及 Stanford 后面的两个空格与 Knuth 后面的一个空格有相同的作用. 而第三句的 TeX 后面的留空就与 Stanford 以及 Knuth 后面的留空不同了, 由此可看出 \ 的作用. 但是汉字与西文之间的空格在第二与第三句出现不同的结果, 可见接在汉字后面的空格与接在西文后面的空格是有区别的.

2.2.3 引号、连字号、破折号

在文稿中输入引号时, 左单引号用键盘上的倒引号 “`”, 亦称重音号, 在键盘打字区左上角, 右单引号直接用键盘上的单引号 “’”.

左双引号是连用两个左单引号, 右双引号是连用两个右单引号或直接用键盘上的双引号 “””, 它与单引号在同一个键上, 是上档键.

当单引号与双引号相邻时, 中间应插入一个小间隔, 命令 “\,” 产生的间隔就很合适. 例如输入 “`\, ‘single’ and ‘double’\,” 产生 “‘single’ and ‘double’”.

使用一个连字号 “-” 产生一个连字号 “-”.

连用两个连字号 “--” 产生一个表示数字范围的符号 “-”.

连用三个连字号 “---” 产生一个西文破折号 “—”.

例如人名 Harish-Chandra 中间是连字符 (hyphen), 而 pages 2-10 中间则是表示数字范围的短线.

为了得到数学中的负号或减号 “-”, 需用数学模式输入成 “\$-\$”.

利用中文输入法, 可以输入各种中文标点符号. 天元系统将中文标点符号当作普通汉字处理, 不具有特殊的控制符号功能.

2.2.4 连体字

在印刷的书籍中,有些特殊的字母组合,它们不是被分离的印出来,而是作为一个连体符号排印. \TeX 将字母组合 ff , fi , fl , ffi 以及 ffl 作为连体字显示为 ff , fi , fl , ffi , ffl , 而不是 ff , fi , fl , ffi , ffl . 通常情况下这样处理是很好的,但有时不作为连体字更好,例如单词 shelfful 中的两个 f 按连体字排版时显示为 shelfful , 就不大美观了. 若要将连体字强制分开,可插入命令 “ $\backslash/$ ”, 上述单词应输入成 $\text{shelf}\backslash/\text{ful}$.

有些字母相邻时,相互距离会拉近. 例如 AV 和 Te 等. 插入命令 $\backslash/$ 可取消这种距离的缩短, 例如输入 $\text{A}\backslash/\text{V}$ 和 $\text{T}\backslash/\text{e}$ 得到 AV 和 Te .

2.2.5 句号后的空白

句号圆点有很多用处,其中两个用处一是用在句末表示句子的结束,二是表示缩写. 这两种圆点后面的空格排版出来的空白长度应有所不同,表示句子结束的空白应更长一些. \TeX 确实是在句号后面插入了额外的间隔,但是对于以小写字母结束的缩写, \TeX 系统并不能自动认出它是缩写,仍当成句子结束处理. 为了“通知” \TeX 系统圆点表示缩写,需将原点后的空格改成 “ \sim ” 号或以 “ $\backslash_$ ” 代替通常的空格,即在空格前面加上倒斜线,这时间隔就是正常的了,不会插入额外空白,而且 \sim 还禁止在该处分行. 所以句子 “Prof. Jones read the Phys. Rev.” 应输入成 “Prof. \sim Jones $\backslash_\text{read}\backslash_\text{the}\backslash_\text{Phys.}\backslash_\text{Rev.}$ ”, 其中第一个缩写后的空格改用了 “不可打断的空格”, 因为称呼和姓名不应分在两行,第二个缩写后的空格使用了 “ $\backslash_$ ”, 既表示圆点不是句号又表示必要时该处可以分行.

\TeX 将紧接在大写字母后的圆点看作是一个缩写而不认为是句号. 但有时大写字母后的圆点确实是句号,这时就需要在圆点前加上 “ $\backslash@$ ”, 以通知 \TeX 按句号处理,得到附加的空白. 例如 “We work in ECNU.” 应输入成 “We work in ECNU $\backslash@$ ”.

2.2.6 非英文字母及重音符号

在非英文的西文文字中有一些特殊的字符,可按下表输入:

\oe	$\{\backslash\text{oe}\}$	\OE	$\{\backslash\text{OE}\}$	\ae	$\{\backslash\text{ae}\}$	\AE	$\{\backslash\text{AE}\}$	\aa	$\{\backslash\text{aa}\}$
\AA	$\{\backslash\text{AA}\}$	\o	$\{\backslash\text{o}\}$	\O	$\{\backslash\text{O}\}$	\l	$\{\backslash\text{l}\}$	\L	$\{\backslash\text{L}\}$
\ss	$\{\backslash\text{ss}\}$	\SS	$\{\backslash\text{SS}\}$	\i	\!	\j	\?		

表中的命令都用花括号括起来了, 这样在一个单词中输入特殊字符时, 可以与前后字符紧靠在一起, 而不必用空格来标志命令的结束, 从而在文稿上不会将一个

单词错看成是两个单词. 例如将schließen输入成schlie{\ss}en比schlie\ss en更好看一些.

有些语言的字母带有各种发音记号(重音符号), 以字母o为例, 可如下输入:

ò \`o	ó \'o	ô \^o	ö \"o	õ \~o
ō \=o	ô \.o	ö \u{o}	ö \v{o}	ö \H{o}
ô \r{o}	ôo \t{oo}	o \b{o}	o \c{o}	o \d{o}

从表中可以看出, 对于由非字母组成的重音命令, 可以不使用花括号, 但由字母组成的重音命令就必须用花括号.

如果要在字母i和j的顶上加记号, 应先去掉原有的点, 可通过命令\i和\j得到i和j. 所以î与ĵ是通过\v{\i}和\H{\j}得到的.

如果需要希腊字母或各种数学符号, 请参阅第四章.

§2.3 分组与环境

不同的命令有不同的作用范围, 有些命令只对其后的一个字符起作用, 有些命令对其后的所有文本起作用. 为了扩展或限定命令的作用范围, 常常需要用一对花括号将一些文本括起来, 称为一个“分组”或一个“集团”. 对于那些只对其后的一个字符起作用的命令, 如果其后是一个分组, 则它就对整个分组起作用. 实际上这样的命令就是带参数的命令. 只是当参数为单个字符时, 可不必用花括号把参数括起来. 例如 This is \textbf{bold face} style 输出为 This is **bold face** style, 可见命令\textbf对其后的整个分组起作用, 对其它位置的字符不起作用. 对于那些对其后的所有文本起作用的命令, 即不带参数的命令(有时称为“声明”), 为了限定作用范围, 可将它放在一个分组的内部, 它的功能到该分组的结束符“}”处就不起作用了. 例如{这是\黑中文黑体}的输出效果是: 这是**中文黑体**, 可见命令“\黑”对它前面的字符以及分组外部的字符是不起作用的.

分组时最常见的错误有两个, 一是记混了上述两种命令的作用范围, 二是漏掉了分组结束的花括号.

在L^AT_EX中, 为了对某些文本进行特定格式的排版, 需要把它们放到相应的环境中, 语法是:

```
\begin{环境名}
  文本
\end{环境名}
```

其中开始和结束的环境名必须相同. 在环境内的文本中可以含有其它命令和声明, 通常它们的作用范围终止于环境的结束. 但要特别注意, 天元中文命令不受环境的约束, 只能用分组花括号限定其作用范围.

§2.4 分段

在源文件中的一个空行(连续按两次回车产生一个空行)相当于一个排版分段命令, \TeX 排版输出时, 空行的上方和下方文稿会被排成两个段落, 这两个段落之间并不出现空行. 源文件中连续多个空行与一个空行的作用相同.

如果不愿意在源文件中用空行来控制分段, 则可用分段命令: 在需要分段的地方插入命令 “ $\backslash\text{par}$ ”, 该命令的前后文稿就会被排成两段.

可以控制两段之间的间隔与两行之间的间隔相同或不同, 也可以控制段落首行缩进或不缩进. 习惯上中文文章段落间隔与行间隔相同, 并且首行缩进, 而西文文章通常两段之间有较大间隔且首行不缩进.

§2.5 分行和分页

当对 \TeX 排版输出的结果不甚满意时, 可以适当进行调整.

2.5.1 分行

强制分行的排版命令是

$\backslash\backslash$ 或 $\backslash\text{newline}$

命令后面的文字会被新起一行从头排起. 这与分段不同, 因为分行的文字仍属于同一段落, 不受段落间距

及首行缩进的影响. 上一句突然中断换行就是因为插入了命令 $\backslash\backslash$.

命令 “ $\backslash\backslash$ ” 还可以带有参数, 形如 “ $\backslash\backslash[\text{长度}]$ ” 或 “ $\backslash\backslash*[\text{长度}]$ ”, 用来增加或减少行间隔. 例如 $\backslash\backslash[5\text{cm}]$ 使当前行与新行之间增加 5cm 的间隔, 而 $\backslash\backslash[-1\text{mm}]$ 使当前行与新行之间的间隔(文稿中在此处放置了命令 $\backslash\backslash[-1\text{mm}]$) 减少了 1mm. 我想你已经能看出命令 $\backslash\backslash[-1\text{mm}]$ 的效果了.

当分页位置正好在当前行和新行之间时, “ $\backslash\backslash[\text{长度}]$ ” 与 “ $\backslash\backslash*[\text{长度}]$ ” 就有区别了, 前者相当于变成了一个分页命令, 使当前行与新行放在两页里, 指定的间隔被“吃掉”了不再起作用. 而后者会在当前行之前分页, 使得当前行和新行同时出现

在新的一页里, 并仍保持指定的间隔. 当指定的间隔超出一页的高度时, 当前行出现在新页的顶部, 新行出现在下一页的顶部.

另一个分行命令是“建议分行”的命令

`\linebreak[数字]`

其中数字是0到4之间的一个整数. 数字越大, 建议的力度也越大, 使用数字4时已不是建议分行而是强制分行, 此时命令可以简写为`\linebreak`, 使用其它数字时, 排版系统根据内部设定的不同阈值, 决定是否采纳分行建议. 与`\`、`\newline`有所不同, `\linebreak`形式的分行命令的输出结果是使当前行撑满整行, 行内单词之间的间距可能变得很大. 读者一定会发现上一行排得特别空松, 这就是插入了命令`\linebreak`后产生的效果.

与建议分行命令相反的是“建议不分行”命令

`\nolinebreak[数字]`

同样是数字越大, 建议的力度也越大, 使用数字4(或不带参数)时已不是建议而是强制不允许在当前位置分行了.

有时需要将一些文本整体保持在同一行中, 不允许在中间任何地方分行, 可使用命令

`\mbox{文本}`

注意不要将太多的文本保持在同一行中, 以免出现超长的行使得输出版面难看.

如果排版后某行末尾的一个外文单词使得该行超长, 可在该词内部插入一个或几个`\-`, 这是建议断词的位置, 断词后一个单词分在了两行, 上一行的末尾会自动插入一个连字符.

2.5.2 分页

通常情况下`TeX`会自动分页, 无需人工干预. 必要时可强制在指定位置分页. 类似于分行的`\newline`命令, 强制分页的命令是

`\newpage`

如果该命令出现在两个段落之间, 则这前后两段就被排版到两页, 并且当上一段所在页面不满时, 该页面下部会出现空白. 如果该命令出现在一段之内, 则在排版输出时会在命令所在位置把一段强制变成两段, 然后在这两段之间强制分页.

建议分页或建议不分页的命令是


```
\pagebreak[数字]  
\nopagebreak[数字]
```

介于0与4之间的参数表达了建议的力度,当参数数字是4时,可以省略,此时建议性的命令实际变成了强制性的命令。

用\pagebreak分页与用\newpage分页的效果是不同的。当在两段之间出现\pagebreak命令时,虽然也是在这两段之间进行分页,但当上一段所在页面不满时,会自动增加段落间距以撑满一页。如果该命令出现在一段之内,则在排版输出时会在当前行完成后再分页。

命令\nopagebreak出现在两段之间时禁止在该处分页,出现在一段之内时,禁止在当前行完成后出现分页。

有时稍稍增加当前页的高度就可能避免一个难看的分页,命令

```
\enlargethispage{尺寸}  
\enlargethispage*{尺寸}
```

可使当前页(仅仅是当前页)增加一点高度。不带*号的命令指定的尺寸是当前页可以增加的最大高度,实际增加值可能小于这个指定值。*形式的命令使当前页严格增加指定的高度,因此可能会使行间隔和段落间隔发生少许变化。

§2.6 水平间距、竖直间距

2.6.1 水平间距

命令\, (即倒斜线后紧跟一个逗号)产生一个很小的水平间距,大约是字母M宽度的1/6。如99 999中间的空格就是\,的效果。

产生指定长度的空白可以使用下述命令

```
\hspace{长度}  
\hspace*{长度}
```

两种格式在通常情况下没有区别,都是在文本中间插入水平空白间隔,但是当排版结果恰好使得开始插入的位置是在新行的开始,则不带*号的命令插入的空白会被吃掉(相当于没写这个命令),而带有*号的命令在任何情况下都会插入指定长度的空白间隔。命令中的长度可以是负值,这会使命令后面的文本向后退,当这个负值的绝对值较大时,会使命令后面的文本退到前面文本的左面产生重叠现象。

下面是两个与当前所用字样(font)有关的插入固定长度的命令

`\quad` `\qquad`

命令`\quad`插入相当于当前字样尺寸的空白,例如当前使用10pt字样时,`\quad`相当于`\hspace{10pt}`,而`\qquad`是`\quad`的两倍.例如

“始 中 尾”的空格就是`\quad`与`\qquad`的效果.

有时为了撑满一行,需要插入未知长度的空白,这时可以使用特殊的弹性长度`\fill`,它具有无限伸缩的能力,最短时长度为零,最长时需要多长有多长.

命令`\hspace{\fill}`,简写为

`\hfill`

这个命令会根据排版的结果,在命令两边的文本之间插入需要的空白,以撑满一行.当这个命令位于多行段落之内时,通常是不起作用的,因为每行都是满的,但当它位于单独的一行或接近段落的末尾时,就容易看到它的效果了,这时弹性长度起到了弹簧的作用,使所在行的文本撑满一行.

当`\hfill`位于一行开头或者末尾时,空白会被吃掉,就好像一根弹簧,若一端没有支撑物,另一端就不能顶开其他物体.此时可在空白的一端放上一个没有宽度的文本如空盒子`\mbox{}`作“支撑物”,或者换用带*号的命令`\hspace*{\fill}`.

利用`\hfill`容易使单行文本作到左对齐、右对齐或居中对齐.例如输入

这是靠左对齐的一小段文本`\hspace*{\fill}\`
`\hspace*{\fill}`这是居中对齐的一小段文本`\hspace*{\fill}\`
`\hspace*{\fill}`这是靠右对齐的一小段文本

得到

这是靠左对齐的一小段文本
 这是居中对齐的一小段文本
 这是靠右对齐的一小段文本

对单行或多行文本作对齐处理,有现成的命令和环境,详见第28页和第119页.上例介绍的对齐方法大多用于表格的单元格中,如果要使某一单元格不按着表格环境预定的列格式对齐文本,上述对齐方法就有用武之地了.

命令

`\dotfill` `\hrulefill`

具有类似于`\hfill`的功能,但它们不是产生单纯的空白,而是分别用点线和实线填充空白.例如输入

左端文本`\hrulefill` 中间文本`\dotfill` 右端文本

得到

左端文本_____中间文本.....右端文本

若用虚线填充指定长度的空白,可参见下面一行的输入和输出:

`\makebox[3cm]{\dotfill}` \Rightarrow

在`TeX`中有一个占位命令

``

它精确地占据了文本的宽度但不显示文本的内容.当需要的水平间距恰是某段文本的宽度时,使用这个命令比使用`\hspace`要方便得多,因为它无须测量这段宽度的具体数值.

2.6.2 竖直间距

使用命令

`\vspace{长度}`
`\vspace*{长度}`

可以在两行之间插入竖直间隔.需要注意的是这两个命令不起分行的作用,即不会自动将命令前后的文本分成两行,只有在当前行完成后才在该行下面插入竖直空白间隔.这一行与上一行的行距增大就是由于上一行中间插入了命令`\vspace{1mm}`.

与水平间隔类似,当插入的空白恰在新一页的开始时,不带*号的命令插入的间隔被吃掉,而带*号的命令在任何情况下都会插入指定的空白间距.插入竖直间隔的命令大多用于两个段落之间.命令中的长度也可以是负值,使下一段的位置向上提升.

与`\hfill`对应的是

`\vfill`

即 `\vspace{\fill}`, 当一页不满时, 它会插入足够的空白以撑满一页.

有三个内部定义好的具有弹性高度的命令可以插入竖直空白, 它们按正常值排列从小到大是

```
\smallskip    \medskip    \bigskip
```

它们的定义是

```
\vspace{\smallskipamount}  
\vspace{\medskipamount}  
\vspace{\bigskipamount}
```

其中的三个长度参数由具体的文档版式所给定, 用户也可重新定义, 例如可如下将“小间隔”重新定义为基本长度 2mm, 最多 3mm, 最少 1.4mm:

```
\setlength{\smallskipamount}{2mm plus 1mm minus 0.6mm}
```

为了让读者有个直观印象, 下面画了 4 条线, 它们之间的距离相当于本书的版式所规定的 3 种间隔:

§2.7 与段落有关的距离

2.7.1 首行缩进

段落首行缩进的距离由长度 `\parindent` 决定, 用户可如下改变这个值:

```
\setlength{\parindent}{长度}
```

当上述长度不是零时, 段落的首行会自动缩进. 对于中文文章, 当使用的中文字体与西文字体尺寸协调时, 命令

```
\setlength{\parindent}{2em}
```

可使每段首行缩进两个汉字的距离.

若想使某段首行不缩进, 可在该段开始放上命令 `\noindent`.

需要注意的是每一节的第一段的首行并不会缩进, 为了使第一段能像其他段一样会首行缩进, 可在第一段的开始放上命令

```
\hspace*{\parindent}
```

或更简单地在源文件的导言区放上命令

```
\usepackage{indentfirst}
```


2.7.2 段落间距

两段之间的距离等于行间隔(`\lineskip`)加上长度`\parskip`的值, 如果重设这个值, 通常应设置成弹性长度并使用`ex`作单位以便它随着字号的改变而变化.

2.7.3 伸展行距

行距(`\baselineskip`)是相邻两行基线之间的距离. 所谓基线可粗略地看成是一行文字底部的直线, 例如一行字母`abxy`中的`abx`座落在基线上, 但字母`y`的下伸部分落在了基线下方.

当选定了一种字号, 行距就被自动确定下来了. 如果要伸展行距, 可通过重新设置伸展因子达到目的, 命令格式为:

`\renewcommand{\baselinestretch}{伸展因子}`

其中伸展因子的值是一个十进制小数, 新的行距是原有基本行距乘以这个因子后的值.

上述命令写在导言区时直接对整个文本起作用, 但若放在正文中(即在命令`\begin{document}`之后), 则不会立即起作用, 只有当字号改变时(遇到一条选择字号的命令), 伸展因子才起作用, 而且是从字号命令所在的段落开始起作用, 不影响前面的段落.

如果只想改变行距而不改变字号, 那么只要随便写一个设置字号的命令, 紧接着再写上原有的字号命令即可. 当文稿中未出现字号命令时, 默认是正常大小, 即相当于使用了命令`\normalsize`. 此时若把以后的行距改为“双行”行距但不改变字号, 可同时使用以下两条命令

```
\renewcommand{\baselinestretch}{1.6}
```

```
\large\normalsize
```

其中设置字号的命令`\large`可以换成其他的字号命令. 此例伸展因子用1.6, 排版后的视觉效果较好, 若用2.0, 则显得过宽.

第三章 文字模式

L^AT_EX 把排版情况分成 3 种模式: 段落模式、左到右模式和数学模式. 段落模式就是把文稿分行、分段和分页, 排成需要的版面; 左到右模式是把输入的字符排成从左到右的一行, 无论长短都不分行; 数学模式用于排版数学公式. 我们把前两种模式统称为文字模式放在本章中一起介绍.

为了版面美观和其他需要, 一篇文稿中大多会使用几种不同的字体. 所谓字体就是文字的风格样式. 具有特定大小和外观的一组字母、数字和符号的集合构成一种特定字体的字符集. 在 L^AT_EX 中用于文字模式的默认字体是直立的罗马字体, 在天元中默认的中文字体是宋体.

§3.1 中文字体

天元提供了一些中文控制命令, 用于选择中文字体、字型和大小. 按 L^AT_EX 术语, 所有中文命令都是“声明”, 即它们都不带参数, 一经使用, 就对其后的文本起作用, 作用范围直到遇到另一个同类命令或遇到所在分组的结束符 “}” 为止.

天元定义了 17 个选择中文字体的命令, 每个命令都是由一个倒斜线跟一个中文字组成. 下面列出其中最常用的 4 个命令, 其他命令请参见附录. 命令后面的文字是输出效果.

<code>\宋</code>	这是宋体字 (默认字体)
<code>\楷</code>	这是楷体字
<code>\仿</code>	这是仿宋体
<code>\黑</code>	这是黑体字

改变字形的命令是

<code>\瘦</code>	这是瘦体字
<code>\阔</code>	这是阔体字
<code>\扁</code>	这是扁体字
<code>\正</code>	恢复正常字体

下面是指定字体大小的命令, 个别命令有两个不同的名字. 括号中的数字是正常大小时汉字的点数.

<code>\半(5)</code>	<code>\五(5)</code>	<code>\六(6)</code>	<code>\七(7)</code>	<code>\八(8)</code>	<code>\小(8)</code>
<code>\九(9)</code>	<code>\标(10)</code>	<code>\中(12)</code>	<code>\大(14)</code>	<code>\特(17)</code>	<code>\双(20)</code>
<code>\巨(25)</code>	<code>\叁(30)</code>	<code>\肆(40)</code>	<code>\伍(50)</code>	<code>\陆(60)</code>	

当使用 ttf 字库时(注意要在天元中作适当配置), 可以使用 GBK 大字符集, 文稿中可以同时出现简体字和繁体字. 如果要输出繁体字, 输入也必须是繁体字, 各种冷僻字只要能输入, 就能排版输出. 但目前大字符集仅限于宋体和黑体, 因为还没有其他字体的 ttf 大字符集字库.

§3.2 西文字体

西文字体的基本尺寸是 10pt、11pt 和 12pt, 从下面 3 行可以看出 3 种尺寸的差别:

This is the 10 pt font. ABXYabxy1234

This is the 11 pt font. ABXYabxy1234

This is the 12 pt font. ABXYabxy1234

3.2.1 字体属性

字体尺寸大小仅是字体的一种属性, 在新字体选择方案(NFSS)中, 每种字体有 5 种属性: 编码、族、系列、形状和尺寸. 普通用户一般不会涉及到字体的编码.

族(family)指的是概观样式. L^AT_EX 2_ε 提供了 3 种声明用于选择不同的族:

`\rmfamily` 切换成罗马(roman)字体

`\sffamily` 切换成无衬线(sans serif)字体

`\ttfamily` 切换成打字机(typewriter)字体

形状(shape)指的是倾斜和高矮, 在 L^AT_EX 2_ε 中提供了 4 种声明用于选择不同的形状:

`\upshape` 切换到直立 (upstanding) 字体
`\itshape` 切换到意大利 (*italic*) 斜体
`\slshape` 切换到称为 *slanted* 的斜体
`\scshape` 切换到小体大写 (SMALL CAPS) 字体

系列 (series) 指的是字体的宽度和权重 (黑度), 权重反映了笔划的粗细. 可用的声明有:

`\mdseries` 切换到中等 (medium) 权重
`\bfseries` 切换到黑体 (**bold face**)

上述一些命令称为声明, 就是说这些声明在遇到新的声明之前一直起作用. 为了限定其作用范围, 应把它放在一组内, 即用花括号把声明和受影响的文本括起来, 或者对于很长的文本使用相应的环境

```

\begin{字体属性}
    使用新属性的文本
\end{字体属性}
  
```

其中的字体属性可以是上面任何一种属性声明, 但要去掉前缀 ‘\’.

还有一个非常重要的声明 `\normalfont`, 它把除了字体尺寸以外的所有属性重设成默认值, 即中等权重的直立的罗马字体.

对应于上面的字体声明, 都有相应的字体命令, 这些命令只对命令参数中的文本起作用, 当改变一小段文本或一个单词的字体属性时, 建议使用字体命令而不使用字体声明. 改变属性的命令有:

族: `\textrm{文本}` `\textsf{文本}` `\texttt{文本}`
 形状: `\textup{文本}` `\textit{文本}` `\textsl{文本}` `\textsc{文本}`
 系列: `\textmd{文本}` `\textbf{文本}`
 默认值: `\textnormal{文本}`
 强调: `\emph{文本}`

这些命令中的文本不能位于两个段落中. 上述强调命令将参数中的字体改变成强调字体: 如果当前字体是直立字体, 强调字体就是 *italic* 斜体, 而如果当前字体是斜体 (*italic* 或 *slanted*), 则强调字体就是直立字体. 与强调命令对应的强调声明是 `\em`.

当由斜体切换到直体时, 斜体字符与直体字符之间的间距会显得比通常的字符间距小, 因此应根据不同的字符而插入一些额外的间距 (*dh* 之间应比 *bh* 之间插入更多一点间距). \TeX 用命令 `\/` 加入这个间距, 称为倾斜校正. 强调命令 `\emph`

会自动插入倾斜校正, 而用强调声明 `\em` 时必须人工加入倾斜校正. 如果使用强调命令时不要倾斜校正, 应在倾斜字符后面加上命令 `\nocorr`.

slanted 字体与 *italic* 字体都是倾斜字体, 但两者字形稍有不同, 倾斜度也不同, 前者向右倾斜了高度的 $1/6$, 后者向右倾斜了 $1/4$.

3.2.2 选择字体尺寸

当在文档类选项中指定了字体的基本尺寸后, 可用下面的声明来改变字体的尺寸. 表中命令后面的文本是在基本尺寸为 10pt 的情况下, 相应声明对应的字体大小的示例:

<code>\tiny</code>	5pt, smallest	<code>\scriptsize</code>	7pt, very small
<code>\footnotesize</code>	8pt, smaller	<code>\small</code>	9pt, small
<code>\normalsize</code>	10pt, normal	<code>\large</code>	12pt, large
<code>\Large</code>	14.4pt, larger	<code>\LARGE</code>	17.28pt
<code>\huge</code>	20.74pt	<code>\Huge</code>	24.88pt

如果要限制字体尺寸声明的作用范围, 应把它放到一个分组中, 或放在任何一个环境内部.

请读者注意, 上述字体尺寸声明仅对西文起作用, 为了能使中文同步缩放, 必须同时加上天元的字体大小的命令, 我们在下表中也列出了在默认的 10pt 情形所对应的印刷汉字的近似字号, 供读者参考.

<code>\tiny</code>	<code>\五</code>	<code>\tiny</code> 相当于七号字
<code>\scriptsize</code>	<code>\七</code>	<code>\scriptsize</code> 介于六号、七号之间
<code>\footnotesize</code>	<code>\小</code>	<code>\footnotesize</code> 相当于六号字
<code>\small</code>	<code>\九</code>	<code>\small</code> 相当于小五号字
<code>\normalsize</code>	<code>\标</code>	<code>\normalsize</code> 相当于五号字
<code>\large</code>	<code>\中</code>	<code>\large</code> 相当于小四号字
<code>\Large</code>	<code>\大</code>	<code>\Large</code> 相当于四号字
<code>\LARGE</code>	<code>\特</code>	<code>\LARGE</code> 相当于三号字
<code>\huge</code>	<code>\双</code>	<code>\huge</code> 相当于二号字
<code>\Huge</code>	<code>\巨</code>	<code>\Huge</code> 相当于一号字

在 L^AT_EX 2_ε 中上述声明仅改变字体尺寸而不改变其它属性,但在老版本的 L^AT_EX 2.09 中,上述字体尺寸声明在改变字体大小的同时,还把其它属性重设成默认值,相当于使用了声明 `\normalfont`.

每种字体实际上都有两个尺寸,第一个尺寸表示字体大小,有时简单地称为字号,通常提到字体尺寸时,大多是指字号;第二个尺寸表示行距,也就是两行基线之间的距离,称为字体的自然行距,对于 L^AT_EX 2_ε 实际安装的字体,第二个尺寸总是大于第一个尺寸.通常情况下排版时控制行距的命令 `\baselineskip` 的值就取自这个自然行距.

如果对自然行距不满意,可以随时改变 `\baselineskip` 的值.例如,当前自然行距是 12pt, 命令

```
\setlength{\baselineskip}{20pt}
```

就把行距改成了 20pt, 看起来就是双倍行距的效果.

需要注意的是, `\baselineskip` 的值对整个段落起作用,而且一个段落中只使用一个 `\baselineskip` 的值,如果一个段落中这个命令的值改变了多次,只有最后一次的值对本段起作用.

由于字体尺寸属性含有两个值,所以每当使用了字体尺寸命令,就相当于改变了两个值,除了字体大小改变外, `\baselineskip` 的值也被重置为该字体尺寸对应的自然行距.

如果在一段落中使用了不同尺寸的字体,并且没有使用花括号限制字体声明的作用范围,那么当最后使用的是最大的字体时,可以明显看出整段都是大的行距.如果把最小的字体放在最后,则意味着整段都用最小的行距,此时不必担心前面相邻两行的大字体出现重叠现象,这是因为 T_EX 除了使用 `\baselineskip` 外,还使用两个值 `\lineskiplimit` 和 `\lineskip`. 当相邻两行太靠近时——上一行的底部与下一行的顶部之间的距离(称为行间隔)小于 `\lineskiplimit` 时,就强制这个距离等于 `\lineskip` 的值,从而避免两行重叠.

调整行距时,不推荐直接修改 `\baselineskip` 的值,最好是使用伸展因子 `\baselinestretch`, 其正常值为 1. 实际上真正的行距是

```
\baselineskip×\baselinestretch
```

用户可以用以下的命令

```
\renewcommand{\baselinestretch}{伸展因子}
```

随时改变这个因子的值,但它的新值只有在又一次遇到改变字体尺寸时才起作用.

§3.3 居中

将较短的文本排放在一行正中间, 可以使用 TeX 原有的命令

```
\centerline{文本}
```

使用这个命令时, 如果文本很长, 那么即使在文本中间加入了分行命令, 也不会分行, 这样排出来的行就会超出行宽. 把多行文本居中对齐排版应使用居中对齐环境

```
\begin{center}  
  第一行\\  
  第二行\\  
  .....  
  第末行\\  
\end{center}
```

在上述环境中如果有某一行文本的长度超出了行宽, 它会被自动地分成几行, 分行时绝不断词, 而且每行单词之间的间隔相同, 这些行都是居中对齐的. 由于不进行断词, 还要保持一致的单词间距, 所以当文本中出现很长的单词时, 自动分行后有些行可能不等长. 对于中文, 由于任何两个字之间都允许分行, 所以自动分行后除最后一行外, 其他行通常是充满一行的.

在居中对齐环境中, 可以使用强制分行命令“\\”, 也可以使用“\\[长度]”, 以增加或减少相邻两行之间的行间隔.

在一个分组或环境内部, 还可以使用声明

```
\centering
```

将后继文本居中对齐, 声明的作用到分组或环境结束时为止. 但对某些超宽的行不会自动分行, 只能使用\\强制分行. 所以居中对齐声明的能力不如上述的居中环境.

如果想使文本居左或居右对齐, 可参见第 119 页.

§3.4 参考文献

在科技书籍和论文中常常要列举大量的参考文献, 在正文中通过文献编号进行引用. 当添加或删除一些文献后, L^AT_EX 会重新进行编号并相应更改引用的编号. 参考文献是用下列环境生成的.


```
\begin{thebibliography}{编号样本}
  \bibitem[记号]{引用标志} 文献条目
  \bibitem[记号]{引用标志} 文献条目
  ...
  \bibitem[记号]{引用标志} 文献条目
\end{thebibliography}
```

对于没有可选项记号的条目, 编译后 `\bibitem` 就会生成一个用方括号括起来的编号, 这些编号总是顺序排列的, 当增加或减少文献条目时, 这些编号会自动改变. 由于 \TeX 在第一次编译时, 先遇到引用, 最后才编译文献目录, 因此第一次编译遇到引用时只能用问号 “?” 代替文献编号, 还需要再编译一次, 才能去掉问号, 代以正确的编号. 不过第二次编译时不需要再做天元.

引用标志不可省略, 它不会显示在排印结果中. 在正文中, 使用

```
\cite{引用标志1, 引用标志2, ...}
```

引用相应的一些文献, 在排印结果中, 上述引用命令就变成了被方括号括起来的文献编号. 引用标志可以用字母、数字和除了逗号外的符号组成.

真正的文献信息写在文献条目中, 包含作者、题目、出版社、年代、版本、页码等内容, 不同部分一般使用不同的字体. 如果一行排不下, 后面的行会向右缩进, 缩进的距离等于编号样本的宽度.

编号样本可以是一个数字, 它的位数应是参考文献最大编号的位数, 以使缩进的文本能够对齐. 如果使用了[记号], 该文献不会被编号, 在相当于编号的位置显示记号本身, 在正文中通过引用标志引用, 排版后显示为给定的记号. 当记号的宽度大于编号的宽度时, 应在编号样本处写最宽的记号.

在参考文献环境中, 标题采用英文 “References”, 字体为 “`\Large\bfseries`”, 而且向左对齐. 如果想采用中文标题或别的西文标题, 可把以下命令插在参考文献环境之前.

```
\def\refname{\hfil\大\黑参\quad考\quad文\quad献}
```

前面加上命令 `\hfil` 是为了按照中文的习惯, 让 “参考文献” 居中. 注意这里不能用 `\hfill`

下面的 3-4-1.ty 给出了引用及参考文献环境的例子.

```
% 3-4-1.ty
关于其中细节, 请参见 \cite{lamport, knuth, dbk}.
\def\refname{\hfil\大\黑\quad 参\quad 考\quad 文\quad 献}
\begin{thebibliography}{Lam}
\bibitem[Lam]{lamport}Lamport, L. \textsl{\LaTeX{}} --- A
Document ...}
\bibitem[Knu]{knuth}Knuth, D. E., \textsl{The \TeX{}}book},
...
\bibitem{dbk}中国大百科全书
\end{thebibliography}
```

其打印输出如下所示.

关于其中细节, 请参见 [Lam, Knu, 1].

参 考 文 献

[Lam] Lamport, L. *LT_EX A Document ...*

[Knu] Knuth, D. E., *The T_EXbook*, ...

[1] 中国大百科全书

§3.5 制表位

使用 WORD 时, 可以在一行上设置几个制表位, 每当按一下 tab 键, 光标就跳到下一个制表位, 这对于排列竖向对齐的几列文字是相当方便的. 在 T_EX 中, 用 tabbing 环境实现了类似功能: 左边界自动是一个制表位 (称为第零个制表位), 然后在一行中任何地方使用设位命令 \=, 该处就成为一个制表位, 在后面的行中, 跳格命令 \> 表示排版时跳到下一个制表位. 第零个制表位是默认使用的, 不需要跳格命令. 每一行都用 \\ 结束.

设置制表位时, 为了精确控制列宽, 可以使用 \hspace 命令, 或者利用后面出现的各列中最宽的项设置制表位, 这样的行称为样本行, 它的作用是仅仅设置制

表位, 而不被显示出来, 规定这种样本行末尾不能使用`\\`, 而必须用命令`\kill`结束.

第66页的表格就是使用`tabbing`环境如下输入的:

```
{\楷\ttfamily
\begin{tabbing}
\hspace*{4em} \=article\quad \=文章类\kill
  \>book      \>书籍类\\
  \>report    \>报告类\\
  \>article   \>文章类\\
  \>letter    \>书信类
\end{tabbing}
}
```

在每一行中, 随时可用设位命令`\=`重设或添加(不是插入)制表位. 如果在一行中有足够的跳格命令, 使最后一个跳格命令正好跳到最后一个制表位, 则可在该行最后一个跳格命令之后的任何位置用`\=`添加一个新的制表位. 如果在中间某个(例如第二个)制表位之后加入了设位命令`\=`, 则是重设该制表位的下一个(即第三个)制表位的位置, 应使重设的制表位不超过再下一个(第四个)制表位的位置, 否则会排版成出乎意料的结果.

下面是重设和添加制表位的例子, 上为输入, 下为输出.

```
\begin{tabbing}
这是第一列\quad \= 这是第二列 \\
左列 \> 中列\quad \= 新添第三列 \\
新一列\quad\= 新二列 \> 对齐第三列\\
列1 \> 列2 \> 列3
\end{tabbing}
```

```
这是第一列 这是第二列
左列      中列  新添第三列
新一列  新二列  对齐第三列
列1      列2    列3
```

关于`tabbing`环境的更深入的介绍, 可参见提高篇.
几点说明:

1. TeX 象处理段落一样处理 `tabbing` 环境, 也就是说, 必要时会自动在两行间分页. 该环境中不能使用分页命令. 如果一定要在某处强制分页, 可使用一个变通的方法: 在想要分页的行末加上一个充分大的行间隔, 例如 `\\[15cm]`, 排版输出时, 这个间隔超出了当前页的剩余空间, 于是就在此处分页了, 又因为在 `\\` 后面未加 `*` 号, 所以下一页顶部不会出现空白. 顺便提及, 后面将要介绍的 `tabular` 环境不会自动分页.

2. 在 WORD 中, 当输入的文本很长时, 可能占用几个制表位, 按 `tab` 键会跳到后面还未占用的制表位上. `tabbing` 环境与此不同, 如果前一个跳格命令跳到第 n 个制表位上, 则无论它后面的文本多长, 下一个跳格命令总是跳到第 $n+1$ 个制表位上, 这实际上可能产生回退, 造成文字重叠. 此外如果一行中的跳格命令超过了制表位的个数, 编译时会显示出错信息.

3. `tabbing` 环境不会自动分行, 只有在遇到 `\\` 时才另起一行, 所以输入的文本不可太长.

4. 在 `tabbing` 环境中, 放在一行中的字体命令只对该行起作用, 不影响其他行. 天元中文命令不受此限. 此外与弹性长度 `\fill` 有关的命令没有作用.

5. 关于制表位的深入技巧, 请参看提高篇第 134 页.

§3.6 表格

3.6.1 表格环境

构造表格可使用环境:

```
\begin{tabular}[竖向位置]{列格式}
  第一行\\
  第二行\\
  ..... \\
  第末行
\end{tabular}
```

表格环境就是创建一个小页. 参数的名字已大致表明了其含义, 下面逐个详细说明.

竖向位置 确定表格在竖直方向上与当前外部文本行的相对位置, 默认(即无该可选项时)表格相对于外部基线居中排放. 可取值为:

t 表格顶部与外部基线对齐;

b 表格底部与外部基线对齐.

列格式 指定表格各列的格式. 列格式由若干项组成, 表格的每一列对应于列格式中的一项, 此外还可能含有对应于表格边界和列间分隔线的一些项. 对应于列的项可以是下列值:

l 对应的列的内容靠左对齐;

c 对应的列的内容居中对齐;

r 对应的列的内容靠右对齐;

对应于边界或列间分隔线的格式项有:

I 画单条竖直线;

II 画双条竖直线(两条相距很近的竖直线);

行(表格行) 表格中的每一行都是由若干列组成, 输入时相邻列之间用符号 & 隔开, 列的内容可以是空的, 但列的分隔符 & 符号不能省略, 除非从某一列开始其后各列都是空白而且不画边界竖线, 该列之后的 & 符号才可以省掉.

对应于表格每一行的输入内容都必须用 \\ 结束.

TeX 把每一列的内容当成一个组, 即相当于用花括号把列的内容括起来了, 因此放在列文本中的一些命令如字体字号声明, 其作用将被局限于该行该列之内. 天元不使用这种默认处理, 所以中文命令若没有放在花括号内部, 它就会对各行各列甚至表格之外的文字起作用.

下面是一个把相邻几列当作一列使用的命令:

\multicolumn{列数}{列格式}{文本} 该命令把接下来的指定个数的列组合成单个列, 其宽度等于各列的总宽度, 包括列间距. 列格式指定组合后的单个列的排放格式, 其中必须包含一个(而且只能包含一个)定位参数 l、c 或 r, 可以同时包含竖线 "|". 当指定的列数是 1 时, 该命令可以改变当前行当前列的对齐方式.

下面是几个在表格中画横线或竖线的命令:

\hline 画一条与表格同宽的水平横线. 隐含换行符, 即画线后会自动换行, 所以在该命令后面不要写 \\ . 该命令只能出现在一行的最前面或行结束符 \\ 的后面, 它在刚结束的行的下面画一条横线. 连用两次该命令, 会画出间距很小的两条横线.

`\cline{m-n}` 从第 m 列的开始位置画一条水平横线到第 n 列的结束位置. 该命令必须位于行结束符 `\\` 的后面, 可以多次出现. 命令 `\cline{2-4}` `\cline{6-9}` 会在刚结束的行的下面画左右两条横线, 一条是从第 2 列到第 4 列, 另一条是从第 6 列到第 9 列. 该命令使用全部列宽, 即包括列间隔空白宽度.

`\vline` 画一条竖直线, 其高度等于所在位置的行高. 放在表格环境开始的列格式中的竖线会画出贯穿整个表格的竖直线, 而这里的命令只画出等于行高的竖直线.

3.6.2 样例

输入简单的表格, 只要看看下面的例子就够了, 只有在设计复杂表格时, 才需要了解上一节或提高篇第 135 页有关的知识.

例 1: 这是两个内容相同的表格, 只是字体和表格线有所不同:

姓名	住址	电话
张爱国	中山路 3 号	12345678
王自强	南京西路 10 号	87654321
李立	北京中路 345 号 501 室	

姓名	住址	电话
张爱国	中山路 3 号	12345678
王自强	南京西路 10 号	87654321
李立	北京中路 345 号 501 室	

上面第一个表格输入如下:

```
\begin{tabular}{l|ll}
姓名      & 住址                & 电话\\
\hline
张爱国 & 中山路 3 号          & 12345678 \\
王自强 & 南京西路 10 号      & 87654321 \\
李立   & 北京中路 345 号 501 室 & \\
\end{tabular}
```

上面第二个表格输入如下:

```

\begin{tabular}{|l|l|l|}\hline
{\黑姓名}&{\黑住址} & {\黑电话}\\\hline
张爱国 &中山路 3 号 & 12345678 \\\hline
王自强 &南京西路 10 号 & 87654321 \\\hline
李立 &北京中路 345 号 501 室 & \\\hline
\end{tabular}

```

输入时在原稿上不需要对齐各项,但为了便于检查错误,输入时稍稍对齐一些还是有好处的.另外注意上一表格输入时省略了一个&符号,而下一表格就不能省掉这个&符号,否则右边框线就会缺少一段连不起来了.

如果我们再看一下上面的两个表,会发现由于第一行的栏目标题不居中,影响了美观.这可利用\multicolumn来解决.此外,为了使单名与双名能对齐,我们可在单名中间插入一个全角的空格.全角的空格等于空出一个汉字的位置,可用于汉字的对齐,能起到西文空格无法起到的作用.此外,为了使表格的标题不会因换页而与表身分离,我们可以把标题也当作表格的一行来处理,并利用命令\\[5pt]使其与表身隔开.现在我们把上述表格的输入修改如下:

```

\begin{tabular}{|l|l|l|}
\multicolumn{3}{c}{\中\黑通\quad讯\quad录}\\\hline
\multicolumn{1}{|c|}{\黑姓名}
&\multicolumn{1}{|c|}{\黑住\quad址}
&\multicolumn{1}{|c|}{\黑电 话}\\\hline
张爱国 &中山路 3 号 & 12345678 \\\hline
王自强 &南京西路 10 号 & 87654321 \\\hline
李 立 &北京中路 345 号 501 室 & \\\hline
\end{tabular}

```

排版结果如下:

通 讯 录

姓名	住 址	电 话
张爱国	中山路 3 号	12345678
王自强	南京西路 10 号	87654321
李 立	北京中路 345 号 501 室	

关于表格更深入的技巧和样例,请参看提高篇第135页.

§3.7 脚注

自动编号的脚注命令是

`\footnote{脚注文本, 如含汉字, 应加命令 '\small'}`

该命令应紧接在需要注释的文字后面, 排版输出时会在该处显示一个脚注标记, 并用较小的字体 (`\footnotesize`) 以脚注形式将脚注文本显示在当前页的底部, 并带有同样的脚注标记. 脚注的首行自动缩排, 在一页上的第一个脚注与正文之间有一条水平短线把正文与脚注分隔开来.

脚注的标记是一个上标形式的数字, 它是自动顺序编号的.

本页的脚注¹是如下产生的: 在“本页的脚注”和“是如下产生的”之间插入了命令

```
\footnote{\small这是一个脚注(footnote)例子}
```

由于脚注文本里的中文不受脚注字体尺寸的控制, 所以用“`\small`”显式地指定中文脚注的大小.

在 `article` 类中, 整个文档对自动编号的脚注统一编号. 在 `report` 和 `book` 文档类, 只在一章内统一编号, 每当开始新的一章, 自动编号的脚注都重新从 1 开始计数.

脚注只能位于通常的段落模式中, 不能位于数学模式、表格、LR 盒子或子段盒子 `\parbox` 中. 但可以位于小页环境中, 此时脚注显示在小页的底部.

关于脚注的更深入的技巧, 请参看提高篇第 140 页.

¹这是一个脚注(footnote)例子

第四章 数学公式

前面几章介绍的是正文的排版(称为文本模式),本章介绍数学公式的排版(数学模式), \TeX 最具竞争力的特点就是能够排版输出精美的数学公式.

为了能充分使用 \LaTeX 提供的数学符号和数学黑体字体,可在源文件导言区放上下列命令

```
\usepackage{latexsym,bm}
```

以加载宏包 `latexsym` 和 `bm`.

§4.1 概述

文本模式和数学模式对排版的要求是不同的,最简单的例子就是在文本模式中通常将字母排成正体,空格可以分隔单词,而在数学模式中则将表示变量的字母排成斜体,空格都被吃掉, \TeX 会自动安排公式各部分之间的间距.

为了标明源文件中某段内容是数学公式,必须在该段内容的两边加上特殊标记,以“通知” \TeX 系统对标记的内容按数学模式进行排版处理.需要特别注意的是这种标记总是“成对”出现的,前一个标记表示进入数学模式,后一个标记表示退出数学模式,前后标记的不匹配是造成 \TeX 排版错误的常见原因.

根据数学公式出现的不同位置,将它们分为两类:出现在一行文字内部的称为行内公式,亦称正文公式(text formulas);出现在两行之间的称为行间公式,亦称显示公式(displayed formulas).

在数学模式中有4个基本命令控制字体的大小式样,它们是:

```
\displaystyle    \textstyle  
\scriptstyle     \scriptscriptstyle
```

可分别称为“显示式样”,“正文式样”,“角标式样”和“二级角标式样”. \TeX 会根据不同情况自动调用合适的式样命令,用户也可以使用这些命令强制改变字体大小.

在数学公式中出现的下述数学符号

+ - / = < > () [] | ' ! :

都可直接从键盘输入,但是数学公式中的花括号(亦称曲括号或大括号)不能直接输入,而必须输入成`\{`和`\}`,这是因为不带前导斜线的花括号是 \TeX 的分组(集团)标记,排版后不会被显示出来.此外注意,如果冒号不作为数学符号,而仅是作为标点符号,则不要直接用符号“:”,而应用命令`\colon`,否则在输出的冒号前面可能产生多余的空白间隔.

数学模式中省略号有如下4种:

输出字符	⋮	⋱
输入序列	<code>\ldots</code>	<code>\cdots</code>	<code>\vdots</code>	<code>\ddots</code>

两个水平的省略号的区别是输出位置不同,一个靠下,另一个上下居中.其它两个多用于矩阵,但没有形如 \cdot^{\cdot} 的上升省略号,如果要用的话,需自己定义,例如定义为:

```
\newcommand{\adots}{\mathinner{\mkern2mu%
\raisebox{0.1em}{.}\mkern2mu\raisebox{0.4em}{.}%
\mkern2mu\raisebox{0.7em}{.}\mkern1mu}}
```

在数学模式中,输入的空格不会输出空白,如果要插入空白间隔,可使用下列控制命令:

```
\quad = 两个空铅(quad)宽度, 即| |
\quad = 一个空铅宽度, 即| |
\; = 5/18 空铅宽度, 即|
\: = 4/18 空铅宽度, 即||
\, = 3/18 空铅宽度, 即||
\! = -3/18 空铅宽度, 即||
\_ = 一个空格, 即| |
\hspace{长度}
\phantom{文本}
```

上述这些命令除了`\;`, `\:`和`\!`只能用于数学模式外,其它命令既可以用于数学模式也可以用于文字模式.其中`\!`为负的空白,实际是缩短间隔.命令`\hspace`和`\phantom`的解释见§2.6.1节.

在数学模式中,字母或文字被看成是变量的名字,显示为数学斜体(`\mathit`),空格均被吃掉不起作用,例如输入`$This is a word$`,显示结果是 $Thisisaword$.

在数学模式中，需要显示普通文本时，应把文本放在 LR 盒子中，即输入成 `\mbox{文本}`，其中文本既可以是西文也可以是汉字。盒子内部的空格是起作用的，但在盒子与数学公式之间，空格不起作用，仍需用上述的一些命令插入适当的间隔。

§4.2 行内公式

出现在一行之内的数学公式称为行内公式，例如“勾股定理 $a^2 + b^2 = c^2$ 也称商高定理”这句话中出现的数学公式就是行内公式。

L^AT_EX 提供的行内公式的标记是 `\begin{math}` 和 `\end{math}`，分别表示行内公式的开始和结束，也可以使用简化的标记 `\(` 和 `\)`，或者更简单的使用 T_EX 本来的标记——开始和结束标记都写作 `$`。例如下列 3 种输入

勾股定理 `\begin{math} a^2+b^2=c^2 \end{math}` 也称商高定理

勾股定理 `\(a^2+b^2=c^2 \)` 也称商高定理

勾股定理 `$ a^2+b^2=c^2 $` 也称商高定理

产生同样输出：

勾股定理 $a^2 + b^2 = c^2$ 也称商高定理

在 L^AT_EX 中将形如

`\begin{...}`

.....

`\end{...}`

的排版标记称为“环境”。对于很短的公式，大多数人喜欢使用简化的标记，对于很长的行内公式建议使用环境标记，并将环境的开始和结束标记分别单独写在一行如下面的样子以便于检查错误和修改文件：

勾股定理

`\begin{math}`

$a^2+b^2=c^2$

`\end{math}`

也称商高定理

§4.3 行间公式

位于两行之间的公式称为行间公式, 例如下面的公式

$$\int_a^b f(x) dx = \lim_{\|\Delta x_i\| \rightarrow 0} f(\xi_i) \Delta x_i$$

就是行间公式. 行间公式有多种情况, 例如公式可能只有一行, 也可能有多行; 行间公式可以带编号, 也可没有编号; 带编号时既可以用人工编号, 也可以由 L^AT_EX 自动编号(这是 L^AT_EX 的主要优点之一). 我们以表格形式列出在 4 种情况下 L^AT_EX 所使用的行间公式的标记.

	无编号	自动编号
单行公式	$\begin{array}{l} \backslash\mathrm{begin}\{\mathrm{displaymath}\} \\ \text{(公式内容)} \\ \backslash\mathrm{end}\{\mathrm{displaymath}\} \end{array}$	$\begin{array}{l} \backslash\mathrm{begin}\{\mathrm{equation}\} \\ \text{(公式内容)} \\ \backslash\mathrm{end}\{\mathrm{equation}\} \end{array}$
多行公式	$\begin{array}{l} \backslash\mathrm{begin}\{\mathrm{eqnarray*}\} \\ \text{(公式内容)} \\ \backslash\mathrm{end}\{\mathrm{eqnarray*}\} \end{array}$	$\begin{array}{l} \backslash\mathrm{begin}\{\mathrm{eqnarray}\} \\ \text{(公式内容)} \\ \backslash\mathrm{end}\{\mathrm{eqnarray}\} \end{array}$

注意, 无编号和自动编号的多行公式的环境仅相差一个星号, 但单行公式的两个环境所用文字是很不同的.

类似于行内公式的 `math` 环境, 行间公式的 `displaymath` 环境也有两个简化形式:

`\[..... \]`

和

`$$ $$`

其中后者是原始 T_EX 本身就有的. 注意, 当使用 `$` 作为数学模式的标记时, 在行内公式两端用的是单个的 `$`, 而在行间公式两端使用的是紧连在一起的双 `$`.

对于自动编号的公式, 也可以人工干预(改变)编号, 或指定多行公式中的某行公式不放编号, 在后面几节中会给出相应的例子. 进一步讨论可参见提高编. 如果不想利用 L^AT_EX 的自动编号功能, 而由人工直接给出编号, 可以使用 T_EX 中原有的格式:

`$$ 数学公式 \eqno 编号 $$`

或

`$$ 数学公式 \leqno 编号 $$`

其中`\eqno`将编号放在公式的右面, `\leqno`将编号放在公式的左面. 对于这种带有人工编号的公式, 不能将`$$\cdots$$`换成`\[\cdots\]`.

例如输入

```
$$a^2+b^2=c^2, \eqno(**) $$
```

由公式(`$$**$`)就可得到所需结论.

可得到以下输出:

$$a^2 + b^2 = c^2, \quad (**)$$

由公式(`(**)`)就可得到所需结论.

输入数学公式时, 数学环境前后的空行仍表示分段. 如果环境后面没有空行, 则行间公式下方的文字不会缩进, 即不形成新的段落.

§4.4 上标和下标

上标和下标统称角标, TeX 系统会用较小的字体排版角标, 并升高上标或降低下标的位置. 上标命令是`^{\dots}`, 下标命令是`_{\dots}`, 当角标是单个字符时不用花括号. 在 §1 的例子中已见到上标的打法, 下面再举几例. 输入

```
\[
  x_1,  x_1^2, x^2_1,  x_{22}^{(n)},  {}^*\!\!x^*
\]
```

排版输出为

$$x_1, x_1^2, x_1^2, x_{22}^{(n)}, {}^*x^*$$

从上例可见, 当有上标和下标时, 输入次序是无关紧要的, `x_1^2`和`x^2_1`输出同样结果 x_1^2 . 另外可见, 公式中间输入的空格不起作用. 本例最后一个例子给出了在一个变量左右两边打印角标的一种方法.

输入多层角标时, 必须注意分组括号的使用, 输入

```
\[
  x^{\{y^z\}}, \quad x^{\{y_z\}}, \quad x^{\{x^{\{x^x\}}\}}
\]
```

输出为

$$x^{y^z}, \quad x^{y_z}, \quad x^{x^{y^x}}$$

输入时若漏掉分组括号, 比如把 $x^{\{y^z\}}$ 输入成 $x^y{}^z$, 则会出现编译错误:

! Double superscript.

如果忽略这个错误, 那么输出结果是 x^{y^z} 而不是 x^{y^z} .

可以看出一级角标字体变小了, 二级角标字体更小, 但更高级的角标不再变化, 都和二级角标一样大. 若当前字体大小是 10pt, 则一级角标为 7pt, 二级角标为 5pt. 因此当汉字用作上下标时, 应加上改变大小的命令 “\七” 或 “\五”, 并且不要忘记将它们括在 $\mbox{\}$ 内. 例如输入

```
\[
  S_{\mbox{\七外}}, \quad S_{V_{\mbox{\五极大}}}
\]
```

输出为

$$S_{\text{外}}, \quad S_{V_{\text{极大}}}$$

当角标位置看起来不明显时, 可以强制改变角标字体的大小或角标的层次(如把一级角标改为二级角标). 下例是变量 y 带有大写字母下标 N , 不同输入产生的不同输出. 输入

```
\[
  y_N, \quad y_{-N}, \quad y_{-\scriptstyle N}
\]
```

输出为

$$y_N, \quad y_N, \quad y_N$$

第一种输出结果几乎看不出是下标, 第二种是把一级下标改为二级下标(字体自动变为二级角标大小), 第三种也是把一级下标改为二级下标, 但强制字体为一级角标的大小, 后两种显然比第一种好看得多.

求导符号的撇号可用控制命令 \prime , 由于撇号应出现在上标位置, 所以需要使用上标命令. 或者直接用键盘符号 ' 代替 \prime , 但因键盘上的撇号(单引号)显示的位置已经在上标的位置了, 因此对它就不要再用上标命令了. 例如输入 $f^{\prime\prime}$ 或 f'' , 输出都是 f'' .

排版双行或多行下标见第 51 页和第 146 页.

§4.5 分式

当输入较短的分式时, 最简单的方法是使用斜线, 如输入 $\$(x+y)/2\$$ 产生输出 $(x+y)/2$. 但本节所讲的分式是指带有水平分数线的分式, 所用命令为

```
\frac{分子}{分母}
```

当分子或分母是单个字符时, 可不用花括号括起来. 下面是一些例子.
输入行内分式

```
分式  $\frac{x+y}{2}$  和  $\frac{1}{1+\frac{1}{2}}$ .
```

输出为

```
分式  $\frac{x+y}{2}$  和  $\frac{1}{1+\frac{1}{2}}$ .
```

输入行间公式

```
\[
  \frac{x+y}{2}, \quad \frac{1}{1+\frac{1}{2}}
\]
```

则输出

$$\frac{x+y}{2}, \quad \frac{1}{1+\frac{1}{2}}$$

行内公式中的分式显得比行间分式小, 这是因为行内分式使用的是角标字体, 分式中的分式使用更小一级的字体 (到二级角标字体为止, 不会再小了). 可以人工改变分子和分母所用字体的大小, 见下例的输入和输出, 其中定义几个控制命令是为了简化输入:

```
\newcommand{\D}{\displaystyle}
\newcommand{\DF}[2]{\frac{\D#1}{\D#2}}
```

输入行内公式 $\$\DF{x+y}2\$$ 输出 $\frac{x+y}{2}$

对于连分式, 所有分子和分母具有同样大小时比较好看, 对比下面两例: 输入

```
\[
  a_0+\frac{1}{a_1+
    \frac{1}{a_2+
      \frac{1}{a_3+
        \frac{1}{a_4}}}}
\]
```

输出

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4}}}}$$

输入

```
\[
  a_0+\DF{1}{a_1
    +\DF{1}{a_2
      +\DF{1}{a_3
        +\DF{1}{a_4}}}}
\]
```

输出

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4}}}}$$

分数线长度的预设值是分子或分母所占的最大宽度, 如果想使分数线变长些, 可在分子或分母两侧添加一些间隔. 在中文文档中, 习惯使用略长的分数线, 可为此定义一个中文分式命令

```
\newcommand{\zwfs}[2]{\frac{\;#1\;}{\;#2\;}}
```

请观察下例中的输入和输出. 输入

```
\[
  \frac{1}{2}\quad \zwfs{1}{2}
\]
```

输出(对比分数线长度)

$$\frac{1}{2} \quad \frac{1}{2}$$

§4.6 根式

打印根式的命令是

开平方: `\sqrt{表达式}`

开高次方: `\sqrt[n]{表达式}`

其中 n 是开方次数. 当被开方表达式是单个字符时可以不用花括号, 若是单个字母, 不用花括号时则需用空格与 `\sqrt` 隔开. 根式可以嵌套. 下面举几个例子.

输入 `\sqrt{2}<\sqrt[3]{3}` 输出 $\sqrt{2} < \sqrt[3]{3}$

输入

```
\[
  \sqrt{a}+\sqrt{b}+\sqrt{c},\quad
  \sqrt{1+\sqrt[p]{1+\sqrt[q]{1+a^2}}}]
\]
```

输出

$$\sqrt{a} + \sqrt{b} + \sqrt{c}, \quad \sqrt{1 + \sqrt[p]{1 + \sqrt[q]{1 + a^2}}}$$

从上面的例子可以看出两个问题, 一是当被开方表达式是高度不同的字母时(如上例中的 a 和 b), 根号上面的横线不在同一水平线上. 为了克服这一缺点, 可以在被开方表达式中插入一个只有高度没有宽度的“数学支柱”(`\mathstrut`), 将被开方表达式支撑到同一高度, 根号上面的横线就位于同一水平线上了, 如下例所示, 输入:

```
\[
  \sqrt{\mathstrut a} + \sqrt{\mathstrut b}
  + \sqrt{\mathstrut c}
\]
```

输出为

$$\sqrt{a} + \sqrt{b} + \sqrt{c}$$

第二个问题是当被开方表达式较高时, 开方次数的位置显得低了些. 一个简单但不完美的调整办法是把开方次数变为上标, 并拉近与根号的水平距离, 即将开方命令中的[n]改为 $[\overset{n}{\!}]$, 如下所示, 输入

```
\[
  \sqrt{1+\sqrt{1+\sqrt{1+a^2}}}\]
\]
```

输出为

$$\sqrt{1 + \sqrt[p]{1 + \sqrt[q]{1 + a^2}}}$$

§4.7 求和、积分

产生求和号和积分号的命令分别是 $\backslash\text{sum}$ 和 $\backslash\text{int}$, 它们有一些共同的特点: 一是通常都带有上下限(与上下标的输入方法相同), 二是符号的大小在行内公式和行间公式中是不同的, 被称为是“具有两种尺寸的数学符号”. 对于求和号, 上下限的位置也会变化. 例如输入

在行内公式中的求和号与积分号: $\backslash\text{sum}_{k=1}^n, \backslash\text{int}_a^b$
 在行间公式中的求和号与积分号: $\backslash[\backslash\text{sum}_{k=1}^n, \backslash\text{int}_a^b\backslash]$

输出为

在行内公式中的求和号与积分号: $\sum_{k=1}^n, \int_a^b$
 在行间公式中的求和号与积分号:

$$\sum_{k=1}^n, \int_a^b$$

在带有上下限的符号后面紧跟着写上 $\backslash\text{limits}$, 然后再写上下限, 就会使上下限的位置移到符号的头顶和脚下. 输入如下内容

在行内公式中的求和号与积分号(注意上下限位置):

`\sum\limits_{k=1}^n, \int\limits_a^b`

输出为

在行内公式中的求和号与积分号(注意上下限位置): $\sum_{k=1}^n, \int_a^b$

与`\limits`对应的命令`\nolimits`则总是强制上下限位于符号的右侧, 例如输入

`\[`
`\sum_{i=1}^n a_i \quad \sum\nolimits_{i=1}^n a_i`
`\]`

输出为

$$\sum_{i=1}^n a_i \quad \sum_{i=1}^n a_i$$

在输入完整的积分公式时, 有两个细节需要注意, 一是被积表达式与其后的微分算子之间最好有一点小的间隔, 即放入小间隔控制符“\,”; 二是微分算符“d”应是直体, 即把它输入成`\mathrm{d}`. 对比下例输出的两个积分公式的细微区别. 输入

`\[`
`\int_a^b f(x)dx \quad \int_a^b f(x)\,,\mathrm{d}x`
`\]`

输出为

$$\int_a^b f(x)dx \quad \int_a^b f(x) \, \mathrm{d}x$$

§4.8 数学重音符号

数学模式中的重音符号在用途与打法上都不同于文字模式的重音符号(参见第15页), 数学中的重音符号通常用于区分同一个字母表示的不同的变量, 输入方法如下:

\hat{a}	<code>\hat{a}</code>	\check{a}	<code>\check{a}</code>	\breve{a}	<code>\breve{a}</code>
\tilde{a}	<code>\tilde{a}</code>	\bar{a}	<code>\bar{a}</code>	\vec{a}	<code>\vec{a}</code>
\acute{a}	<code>\acute{a}</code>	\grave{a}	<code>\grave{a}</code>	\mathring{a}	<code>\mathring{a}</code>
\dot{a}	<code>\dot{a}</code>	\ddot{a}	<code>\ddot{a}</code>		

在数学模式中当字母 i 和 j 带有重音符号时, 应去掉顶上的小点, 此时这两个字母需改用 `\imath` 和 `\jmath`, 例如为了输出 \hat{i} 与 \breve{j} , 应输入 `$(\hat{\imath})$` 与 `(\breve{j})`.

符号 `\hat` 和 `\tilde` 有“宽型”版本 `\widehat` 和 `\widetilde`, 它们可伸展到3到4个字母的宽度, 例如输入

```
\[
\widehat{ab} + \widehat{cdef} = \widetilde{xyz}
\]
```

输出为

$$\widehat{ab} + \widehat{cdef} = \widetilde{xyz}$$

§4.9 上划线、下划线及类似符号

在公式的上方和下方划直线的命令分别是

```
\overline{公式}    \underline{公式}
```

在公式的上方和下方划花括号的命令分别是

```
\overbrace{公式}    \underbrace{公式}
```

看几个例子, 输入:

```
\[
\overline{\overline{a}^2 + \underline{ab} + \bar{b}^2}
\]
```

输出为

$$\overline{a^2 + \underline{ab} + \overline{b^2}}$$

输入

```
\[
  \underbrace{a + \overbrace{b + \dots + b}
^{\scriptstyle m\mbox{\scriptsize\七个}}} + c}_{\scriptstyle 20\mbox{\scriptsize\七个}}
\]
```

输出为

$$\underbrace{a + \overbrace{b + \dots + b}^{m\uparrow} + c}_{20\uparrow}$$

从例子中可以看出对于单个字符使用 `\overline` 比使用 `\bar` 得到的线要稍长一些。此外, 用于上下花括号的角标总是出现在花括号的正中上方或下方, 而不是通常的角标位置。

§4.10 堆叠符号

当需要把一个符号堆叠在另一个符号的上方时, 可以使用命令

```
\stackrel{上位符号}{基位符号}
```

其中基位符号会被用正常字体打印在正常位置, 而上位符号则是用较小的字体打印在基位符号的上方, 例如输入

```
\[
  \vec{x} \stackrel{\mathrm{def}}{=} (x_1, \dots, x_n)
\]
```

输出为

$$\vec{x} \stackrel{\mathrm{def}}{=} (x_1, \dots, x_n)$$

利用控制数学字体大小的命令可以使堆叠的上下符号具有同样大小, 对比下例中的三个“小于或等于”符号. 输入

```
\[
  A \le B
  \stackrel{<}{=} C
  \stackrel{\textstyle<}{=} D
\]
```

输出为

$$A \leq B \leq C \leq D$$

其中第一个是使用现成的命令 `\le`, 第二个是使用堆叠命令构造的小于等于号, 第三个也是使用堆叠命令构造的新符号, 但将上位符号的大小改成了与基位符号大小相同.

`\stackrel` 命令的两个参数处于不平等地位, 如果想打印一个类似于分式但不带分数线的公式, 就不能使用这个命令了, 此时可以使用后面将要介绍的 `\begin{array}` 环境, 或使用 TeX 原有的数学命令:

```
{上位公式 \atop 下位公式}
{上位公式 \choose 下位公式}
```

这两个命令中的上位公式与下位公式地位是平等的, 使用同样大小的字体, 它们都生成一个类似于没有分数线的分式结构. 其中 `\choose` 生成的整个公式被包围在圆括号内, 是二项式系数的一种表示方法. 例如输入

```
\[
  \{n+1 \choose k\} = \{n \choose k\} + \{n \choose k-1\}
\]
```

输出为

$$\binom{n+1}{k} = \binom{n}{k} + \binom{n}{k-1}$$

使用 `\atop` 命令打印双行下标很方便, 例如输入

```
\[
  \sum_{k_0,k_1,\ldots>0} \atop k_0+k_1+\cdots=n\}
  A_{0k_0}A_{1k_1}\cdots
\]
```

输出为

$$\sum_{\substack{k_0,k_1,\ldots>0 \\ k_0+k_1+\cdots=n}} A_{0k_0}A_{1k_1}\cdots$$

上例还演示了两种水平省略号的用法, 在逗号后用的是 `\ldots`, 在加号后用的是 `\cdots`. 一般来说, 水平省略号应与它前面的整体符号的中部对齐, 由此决定使用哪一种省略号.

若排版多行下标, 见第 146 页.

§4.11 可以变大的定界符

这里所谓的定界符是指包围或分割公式的一些符号, 如

(())	[\lfloor]	\rfloor
[[]]	[\lceil]	\rceil
{	\{	}	\}	<	\langle	>	\rangle
			\	↑	\uparrow	↑	\Uparrow
/	/	\	\backslash	↓	\downarrow	↓	\Downarrow
				↕	\updownarrow	↕	\Updownarrow

在上述符号前加上命令 `\big`、`\Big`、`\bigg` 或 `\Bigg`, 这些符号就会被放大:

正常大小	() [] { } [] [] < > / \ ↑ ↑ ↓ ↓ ⇅ ⇅
\big	() [] { } [] [] < > / \ ↑ ↑ ↓ ↓ ⇅ ⇅
\Big	() [] { } [] [] < > / \ ↑ ↑ ↓ ↓ ⇅ ⇅
\bigg	() [] { } [] [] < > / \ ↑ ↑ ↓ ↓ ⇅ ⇅
\Bigg	() [] { } [] [] < > / \ ↑ ↑ ↓ ↓ ⇅ ⇅

上述命令放大固定的倍数: 1.5 倍、2 倍、2.5 倍、3 倍, 此外还有自适应的放大命令 `\left... \right`, 它们会根据定界符所包围的公式的大小自动变大, 见下例的括号. 输入

```
\[
  \Bigg( \sum_{k=\frac{1}{2}}^{N^2} \Bigg) \quad
  \left( \sum_{k=\frac{1}{2}}^{N^2} \right)
```

输出为

$$\left(\sum_{k=\frac{1}{2}}^{N^2} \right) \quad \left(\sum_{k=\frac{1}{2}}^{N^2} \right)$$

其中即使是放大倍数最大的 `\Bigg` 也仍显得括号小了点, 而 `\left... \right` 会随着所围公式的变化而变化.

`\left... \right` 总是成对出现的, 缺一不可. 此外还需要注意两点: 一是左右两个符号不能排版到两行上, 即中间不能插入换行符; 二是如果只需在公式一侧有自动变化大小的定界符, 那么只要用英文句号 (即小圆点) 代替另一侧那个不出现的定界符即可, 打印时这个圆点是不会出现的, 例如输入

```
\[
  \left. \frac{\partial f(x,y)}{\partial x} \right|_{x=0}
```

输出为

$$\left. \frac{\partial f(x, y)}{\partial x} \right|_{x=0}$$

放大左括号时, 可以将 `\big, \dots, \Bigg` 等写成 `\bigl, \dots, \Biggl`, 放大右括号时写成 `\bigr, \dots, \Biggr`, 在 L^AT_EX 中它们没有新的作用. 但是如果写成 `\bigm, \dots, \Biggm`, 则被放大的符号就被当成是关系运算符来处理, 即这个符号两侧会插入空白间隔, 对比下例竖线两边的空白. 输入

```
\[
  a \big| b   \quad \quad a \bigm| b
\]
```

输出为

$$a|b \quad a \big| b$$

§4.12 矩阵

使用阵列 (`array`) 环境可以把一些元素排列成横竖都对齐的矩形阵列, 命令格式是

```
\begin{array}[竖向位置]{列格式}
  第一行\\
  第二行\\
  .....\\
  第末行
\end{array}
```

其中的参数含义与 `tabular` 表格环境的参数含义完全相同, 每行的输入格式也是相同的, 例如一行中的各列是用 `&` 符号分隔的, 每行都用 `\\` 结束 (最末行不必写换行符), 详见第 32 页及其后的内容. 但这个阵列环境不能用于文字模式, 只能用于数学模式, 换言之, 阵列环境必须位于数学环境之内, 或者用一对 `$` 或一对 `$$` 括起来. 如果是排版矩阵, 左右应使用可以自动调整大小的圆括号或方括号.

下面是一个简单的矩阵例子. 输入

可如下输入:

```
\[
  f(x)=\left\{
    \begin{array}{ll}
      x & \&\mbox{当 } \$x\ge 0\$ \text{ 时;}\\
      -x & \&\mbox{其他情形.}
    \end{array}
  \right.
\]
```

注意数学公式里出现的汉字必须放在 `\mbox{}` 内, 而在 `\mbox{}` 的内部仍可出现数学模式.

§4.13 单行公式与多行公式

单行公式和多行公式显然是指行间公式. 自动编号的单行公式环境是

```
\begin{equation}
  公式
\end{equation}
```

不参与编号的单行公式环境是

```
\begin{displaymath}
  公式
\end{displaymath}
```

可以简写成

```
\[ 公式 \]
```

人工编号的单行公式可使用 TeX 原有的行间公式标记

```
$$ 公式 \eqno 编号 $$    或    $$ 公式 \leqno 编号 $$
```

上面几处提到的公式既可以是普通的单行公式, 又可以是作为一个整体处理的环境或盒子, 例如是一个 `array` 环境.

如果要改变公式自动编号的值, 需在该公式之前用如下命令设置公式编号计数器的初值:

```
\setcounter{equation}{数}
```

其中的数应是整数. 当该命令位于环境之外时, 其后环境中第一个编号公式的序号值比计数器的值多1.

为了以后引用自动编号的公式, 可用\label{公式标记}命令为这个公式加个标记, 以后就用\ref{公式标记}命令来引用此公式. 不过为了得到正确的编号, 有时还需要用TeX再编译一次. 例如输入

```
\setcounter{equation}{5}
\begin{equation}\label{squaresum}
    a^2+b^2=c^2,
\end{equation}
由公式(\ref{squaresum})就可得到所需结论.
```

可得到以下输出:

$$a^2 + b^2 = c^2, \quad (6)$$

由公式(6)就可得到所需结论.

多行公式就是包含几行的公式, 所有行都在一个特定位置上下竖直对齐, 这个对齐位置大多是一个关系符号所在位置. 实现这一功能的环境有两个, 一是标准的, 一是带*号的:

```
\begin{eqnarray}
    第一行\\
    第二行\\
    .....\\
    第末行
\end{eqnarray}
```

```
\begin{eqnarray*}
    第一行\\
    第二行\\
    .....\\
    第末行
\end{eqnarray*}
```

每行的格式是

左边公式 & 中间公式 & 右边公式 \\

也就是说, 每行都有3列, 列间用符号&分开, 每行用\\结束. 在显示时, 所有行的左边公式在左列中是靠右对齐的, 右边公式在右列中是靠左对齐的, 中间公式在中列中是居中对齐的. 因此每一行就像是环境`\begin{array}{rcl}...\end{array}`中的行.

通常中间公式是单个的数学符号, 如关系符号 $=, <, \leq, >, \geq$ 等.

多行公式环境`eqnarray`与阵列环境`array`是不同的, 有如下区别:

1. `eqnarray`环境本身就是数学模式, 所以不要再加数学模式标记. 但`array`环境是一个只能用于数学模式的表格环境, 它本身并不会自动进入数学模式, 必须用数学模式标记把它括起来.

2. `eqnarray`环境被排版成行间公式, 而`array`环境则根据两边的数学模式标记排版成行内公式或行间公式, 但`array`环境内的各列总是按行内公式选取符号的大小尺寸, 例如分数的分子和分母是用角标字体显示的. `eqnarray`环境的左右两列按行间公式选取符号的尺寸, 但中间一列却是按行内公式选取符号的尺寸, 所以中间一列不要放分数, 一定要放的话, 可显式指定使用`\displaystyle`字体.

3. `eqnarray`环境可以自动对每行公式编号, 而`array`环境不能被自动编号, 必要时可显式指定公式的编号.

标准形式(即不带*号)的`eqnarray`环境对每行公式自动编号, 带*号的环境则不参与公式编号. 为了对标准形式中的某行不加公式编号, 应在该行换行符\\之前插入命令

`\nonumber`

下面是一个多行公式的例子.

$$d(uv) = (uv)'dx = (u'v + uv')dx \quad (2)$$

$$= v(u'dx) + u(v'dx)$$

$$= vdu + u dv \quad (5)$$

这样就得到了公式(5).

是如下输入的:


```

{
\setlength{\arraycolsep}{2.5pt}
\setcounter{equation}{1}
\begin{eqnarray}
d(uv) & = & (uv)'dx=(u'v+uv') dx\\
& = & v(u'dx)+u(v'dx) \nonumber\\
\setcounter{equation}{5}
& = & v du+u dv \label{leibmiz}
\end{eqnarray}
这样就得到了公式(\ref{leibniz}).
}

```

本段文本还示例了命令 `\nonumber` 和 `\setcounter` 的用法和效果. 尤其当为计数器置数的命令位于环境内部时, 其后第一个编号公式的序号值取计数器的值.

在阵列环境中, 参数 `\arraycolsep` 表示每个列的两端留空的长度, 其值是列间距的一半, 默认值比较大, 此处改用了较小的值, 使得两列之间不显得太宽.

下面也是一个多行公式, 有意出现瑕疵:

$$\begin{aligned}
 \sum_{i=1}^{20} i &= 1+2+3+4+5+6+7+8+9+10 \\
 &\quad +11+12+13+14+15+16+17 \\
 &\quad +18+19+20
 \end{aligned}$$

这是如下输入的:

```

\begin{eqnarray*}
\sum_{i=1}^{20} i & = & 1+2+3+4+5+6+7+8+9+10 \\
& & +11+12+13+14+15+16+17 \\
& & +18+19+20
\end{eqnarray*}

```

输出的公式很不好看. 首先是行距不同, 这是因为求和号撑大了所在行的总高度. 改进方法是将该行的换行符 `\\` 改成 `\\[-6pt]`, 使下一行向上提升一段距离. 其次是第二行和第三行开始的两个加号既未上下对齐又与后面数字的间距不同, 这是由于加号的两个不同意义造成的. 加号作为单目运算符时, 是正负号标志, 与

其后的运算量之间间隔很小, 作为双目运算符时, 与两边的运算量之间有稍大的间隔. 输入成 `&&+11` 时, 加号显然是单目运算符, 因为前面的 `&` 是列分隔符, 不是运算量. 输入成 `&&\{+18` 时, 加号就成为双目运算符了, 所以只要将 `&&+11` 改为 `&&\{+11` 就能使二、三两行对齐. 修改后显示为

$$\sum_{i=1}^{20} i = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 \\ + 11 + 12 + 13 + 14 + 15 + 16 + 17 \\ + 18 + 19 + 20$$

§4.14 数学字体

4.14.1 选择数学字体

显示变量名时, 默认使用数学斜体. 可以使用的数学字体命令有

```
\mathrm \mathit \mathtt \mathsf \mathbf \mathcal
```

这些命令开始字符都是 `math`, 其后的字母表明了字体形状. 其中最后一个命令 `\mathcal` 是数学花体, 只能用于大写拉丁字母. 使用这些字体命令时, 文本中的空格一律不起作用. 这些字体命令只能用于数学模式中, 例如输入

```
$g=9.8\,,\mathrm{m/s^2}$
```

输出是 $g = 9.8 \text{ m/s}^2$. 下面是各种字体的例子:

输入	输出
<code>\mathrm{ABxy \Gamma 1234+\sum\sin\alpha}</code>	$ABxy\Gamma 1234 + \sum \sin \alpha$
<code>\mathit{ABxy \Gamma 1234+\sum\sin\alpha}</code>	$ABxy\Gamma 1234 + \sum \sin \alpha$
<code>\mathtt{ABxy \Gamma 1234+\sum\sin\alpha}</code>	$ABxy\Gamma 1234 + \sum \sin \alpha$
<code>\mathsf{ABxy \Gamma 1234+\sum\sin\alpha}</code>	$ABxy\Gamma 1234 + \sum \sin \alpha$
<code>\mathbf{ABxy \Gamma 1234+\sum\sin\alpha}</code>	$ABxy\Gamma 1234 + \sum \sin \alpha$
<code>\mathcal{ABxy \Gamma 1234+\sum\sin\alpha}</code>	$AB\mathcal{x}\Gamma 1234 + \sum \sin \alpha$

可见这些命令只影响拉丁字母、数字和大写希腊字母, 不影响数学符号、已定义的函数名称和小写希腊字母. 此外数学花体只应当用于 26 个大写拉丁字母.

4.14.2 希腊字母

凡是字形与拉丁字母相同的希腊字母, 都可直接从键盘输入. 没有对应符号的, 使用下述命令输入:

小写希腊字母:

α	<code>\alpha</code>	θ	<code>\theta</code>	ξ	<code>\xi</code>	τ	<code>\tau</code>
β	<code>\beta</code>	ϑ	<code>\vartheta</code>	π	<code>\pi</code>	υ	<code>\upsilon</code>
γ	<code>\gamma</code>	ι	<code>\iota</code>	ϖ	<code>\varpi</code>	ϕ	<code>\phi</code>
δ	<code>\delta</code>	κ	<code>\kappa</code>	ρ	<code>\rho</code>	φ	<code>\varphi</code>
ϵ	<code>\epsilon</code>	λ	<code>\lambda</code>	ϱ	<code>\varrho</code>	χ	<code>\chi</code>
ε	<code>\varepsilon</code>	μ	<code>\mu</code>	σ	<code>\sigma</code>	ψ	<code>\psi</code>
ζ	<code>\zeta</code>	ν	<code>\nu</code>	ς	<code>\varsigma</code>	ω	<code>\omega</code>
η	<code>\eta</code>						

大写希腊字母:

Γ	<code>\Gamma</code>	Λ	<code>\Lambda</code>	Σ	<code>\Sigma</code>	Ψ	<code>\Psi</code>
Δ	<code>\Delta</code>	Ξ	<code>\Xi</code>	Υ	<code>\Upsilon</code>	Ω	<code>\Omega</code>
Θ	<code>\Theta</code>	Π	<code>\Pi</code>	Φ	<code>\Phi</code>		

希腊字母只能用于数学模式. 大写希腊字母通常是直立罗马字体, 如若需要斜体的大写希腊字母, 可使用命令 `\mathit` 或 `\mathnormal`, 区别是输出的字符间距不同, 例如

`$\mathnormal{\Gamma\Theta\Xi\Omega\Pi}$` 输出 $\Gamma\Theta\Xi\Omega\Pi$.

`$\mathit{\Gamma\Theta\Xi\Omega\Pi}$` 输出 $\Gamma\Theta\Xi\Omega\Pi$.

对于拉丁字母, `\mathnormal` 与 `\mathit` 也有同样的区别, 即后者字符的宽度和间距都稍紧一些.

4.14.3 数学黑体

使用数学黑体命令 `\mathbf` 不能将公式中的所有符号都变成黑体, 为解决这一问题, 可使用字体命令

<code>\boldmath</code>	<code>\unboldmath</code>
------------------------	--------------------------

前者把数学公式中的大写和小写拉丁字母、小写希腊字母和其他符号设成黑体(但仍有一些符号例外, 例如具有两种尺寸的符号), 后者取消前者的作用, 把数学字体重新设成正常字体.

注意这两个命令虽然是数学字体命令,但不能用于数学模式中. 一种用法是在进入数学模式之前使用命令`\boldmath`,退出数学模式之后立即使用`\unboldmath`. 另一种用法是在数学模式内部使用LR-盒子`\mbox{\boldmath$...$}`,将部分公式设成黑体.

如果在导言区使用命令

```
\usepackage{bm}
```

加载`LATEX`宏包,就可以使用命令

```
\boldsymbol{数学符号}
```

将数学符号(包括具有两种尺寸的符号)设成黑体. 如果对整个公式使用这个命令,那么整个公式都是黑体了. 可以说只要使用宏包`bm`和命令`\boldsymbol`,不必再记其它的数学黑体命令了. 最后给出一个例子,输入:

```
\[
  \Delta = \sum_{k=1}^n \delta_k + A_k \sim
  \boldsymbol{\Delta = \sum_{k=1}^n \delta_k + A_k}
\]
```

输出为

$$\Delta = \sum_{k=1}^n \delta_k + A_k \sim \boldsymbol{\Delta = \sum_{k=1}^n \delta_k + A_k}$$

§4.15 数学符号表

绝大多数的数学符号不能直接从键盘上得到,只能使用命令序列产生各种符号. 去掉命令的前缀倒斜线,剩下的字符串基本反映了符号的数学名称. 数学符号多种多样,前面介绍过的希腊字母和数学重音字母也可看成是数学符号. 下面分类列出数学符号和对应的命令,其中标有下划线的命令表示需要事先加载宏包`latexsym`,当然命令本身是没有下划线的. 此外,对个别符号给出了两种命令.

除了具有两种尺寸的符号之外,数学模式内的其他符号和字符都受当前字体尺寸的控制,但是不要将选择字体尺寸的命令放在数学模式内部,否则这种命令不起任何作用.

4.15.1 二元运算符

\pm	<code>\pm</code>	\cap	<code>\cap</code>	\circ	<code>\circ</code>	\bigcirc	<code>\bigcirc</code>
\mp	<code>\mp</code>	\cup	<code>\cup</code>	\bullet	<code>\bullet</code>	\square	<code>\Box</code>
\times	<code>\times</code>	\oplus	<code>\oplus</code>	\diamond	<code>\diamond</code>	\Diamond	<code>\Diamond</code>
\div	<code>\div</code>	\sqcap	<code>\sqcap</code>	\triangleleft	<code>\triangleleft</code>	\bigtriangleup	<code>\bigtriangleup</code>
\cdot	<code>\cdot</code>	\sqcup	<code>\sqcup</code>	\triangleright	<code>\triangleright</code>	\bigtriangledown	<code>\bigtriangledown</code>
$*$	<code>\ast</code>	\vee	<code>\vee</code>	\trianglelefteq	<code>\trianglelefteq</code>	\triangleleft	<code>\triangleleft</code>
\star	<code>\star</code>	\wedge	<code>\wedge</code>	\trianglerighteq	<code>\trianglerighteq</code>	\triangleright	<code>\triangleright</code>
\dagger	<code>\dagger</code>	\oplus	<code>\oplus</code>	\oslash	<code>\oslash</code>	\setminus	<code>\setminus</code>
\ddagger	<code>\ddagger</code>	\ominus	<code>\ominus</code>	\odot	<code>\odot</code>	\wr	<code>\wr</code>
\amalg	<code>\amalg</code>	\otimes	<code>\otimes</code>				

4.15.2 关系运算符

\leq	<code>\le</code> 或 <code>\leq</code>	\geq	<code>\ge</code> 或 <code>\geq</code>	\neq	<code>\ne</code> 或 <code>\neq</code>	\sim	<code>\sim</code>
\ll	<code>\ll</code>	\gg	<code>\gg</code>	\doteq	<code>\doteq</code>	\simeq	<code>\simeq</code>
\subset	<code>\subset</code>	\supset	<code>\supset</code>	\approx	<code>\approx</code>	\asymp	<code>\asymp</code>
\subseteq	<code>\subseteq</code>	\supseteq	<code>\supseteq</code>	\cong	<code>\cong</code>	\smile	<code>\smile</code>
\sqsubset	<code>\sqsubset</code>	\sqsupset	<code>\sqsupset</code>	\equiv	<code>\equiv</code>	\frown	<code>\frown</code>
\sqsubseteq	<code>\sqsubseteq</code>	\sqsupseteq	<code>\sqsupseteq</code>	\propto	<code>\propto</code>	\bowtie	<code>\bowtie</code>
\in	<code>\in</code>	\ni	<code>\ni</code>	\prec	<code>\prec</code>	\succ	<code>\succ</code>
\vdash	<code>\vdash</code>	\dashv	<code>\dashv</code>	\preceq	<code>\preceq</code>	\succeq	<code>\succeq</code>
\models	<code>\models</code>	\perp	<code>\perp</code>	\parallel	<code>\parallel</code>	\mid	<code>\mid</code>
$=$	<code>=</code>	$>$	<code>></code>	$<$	<code><</code>		

最后3个符号可以直接从键盘上输入, 没有对应的命令, 写在此处仅仅是为了强调它们也是关系运算符.

符号 \parallel 与 \mid 作为定界符出现过, 见第51页. 但现在是作为关系运算符出现的, 与作为定界符时的命令是不同的.

关系运算符与两侧的变量之间有一些额外的空白, 而定界符是不带额外空白的, 例如输入 `$a\mid b$` 和 `$a|b$`, 输出结果为 $a \mid b$ 和 $a|b$, 显然是不同的, 使用时不要混淆. 如果要取消关系运算符两侧的额外空白, 只需简单地用花括号把它括起了. 例如输入 `$a=b$`, `$a{=}b$`, `$a{=b}$`, `${a=}b$`, 输出是 $a = b$, $a = b$, $a = b$, $a = b$, 紧靠着花括号那一侧的额外空白不见了.

在关系运算符上划一个斜线就是否定形式的关系运算符, 例如在“ \equiv ”上划一斜线得到“ \neq ”. 在一个符号上划斜线的“通用”方法是在其命令前面加上`\not`. 对于不等号, 由于经常使用, 除了可由`\not=`产生外, 还可以使用上面表中已定义的命令`\ne`或`\neq`. 下表是一些否定形式的关系运算符:

\nless	<code>\not<</code>	\ngtr	<code>\not></code>	\neq	<code>\not=</code>
\nleq	<code>\not\le</code>	\ngeq	<code>\not\ge</code>	\nequiv	<code>\not\equiv</code>
\nprec	<code>\not\prec</code>	\nsucc	<code>\not\succ</code>	\nsim	<code>\not\sim</code>
\npreceq	<code>\not\preceq</code>	\nsucceq	<code>\not\succeq</code>	\nsimeq	<code>\not\simeq</code>
\nsubset	<code>\not\subset</code>	\nsupset	<code>\not\supset</code>	\napprox	<code>\not\approx</code>
\nsubseteq	<code>\not\subseteq</code>	\nsupseteq	<code>\not\supseteq</code>	\ncong	<code>\not\cong</code>
\nsubsetneq	<code>\not\subsetneq</code>	\nsupsetneq	<code>\not\supsetneq</code>	\nasymp	<code>\not\asymp</code>
\ni	<code>\not\ni</code>	\notin	<code>\not\in</code>	\notin	<code>\notin</code>

要特别注意最后两个符号, 在数学上这实际是同一个符号, 只不过是用不同命令生成的, 显然专用命令`\notin`生成的符号 \notin 优于通用命令`\not\in`生成的符号 \notin .

4.15.3 箭头符号

\leftarrow	<code>\leftarrow</code> 或 <code>\gets</code>	\longleftarrow	<code>\longleftarrow</code>	\uparrow	<code>\uparrow</code>
\Lleftarrow	<code>\Lleftarrow</code>	\Longleftarrow	<code>\Longleftarrow</code>	\Uparrow	<code>\Uparrow</code>
\rightarrow	<code>\rightarrow</code> 或 <code>\to</code>	\longrightarrow	<code>\longrightarrow</code>	\downarrow	<code>\downarrow</code>
\Rightarrow	<code>\Rightarrow</code>	\Longrightarrow	<code>\Longrightarrow</code>	\Downarrow	<code>\Downarrow</code>
\leftrightarrow	<code>\leftrightarrow</code>	\longleftrightarrow	<code>\longleftrightarrow</code>	\updownarrow	<code>\updownarrow</code>
\Leftrightarrow	<code>\Leftrightarrow</code>	\Longleftrightarrow	<code>\Longleftrightarrow</code>	\Updownarrow	<code>\Updownarrow</code>
\mapsto	<code>\mapsto</code>	\longmapsto	<code>\longmapsto</code>	\nearrow	<code>\nearrow</code>
\hookleftarrow	<code>\hookleftarrow</code>	\hookrightarrow	<code>\hookrightarrow</code>	\searrow	<code>\searrow</code>
\leftharpoonup	<code>\leftharpoonup</code>	\rightharpoonup	<code>\rightharpoonup</code>	\swarrow	<code>\swarrow</code>
\leftharpoondown	<code>\leftharpoondown</code>	\rightharpoondown	<code>\rightharpoondown</code>	\nwarrow	<code>\nwarrow</code>
\rightleftharpoons	<code>\rightleftharpoons</code>	\leadsto	<code>\leadsto</code>	\iff	<code>\iff</code>

表中的`\Longleftrightarrow`生成的箭头(\Longleftrightarrow)是普通箭头, 而`\iff`生成的是数学中表示等价关系的箭头(\iff), 后者的两侧有额外的空白, 实际是一个关系运算符.

4.15.4 其他符号

为了使相关符号放在一起, 下表中的个别符号与前面表格有少许重复.

\aleph	<code>\aleph</code>	$'$	<code>\prime</code>	\forall	<code>\forall</code>	\Box	<code>\Box</code>
\hbar	<code>\hbar</code>	\emptyset	<code>\emptyset</code>	\exists	<code>\exists</code>	\Diamond	<code>\Diamond</code>
\imath	<code>\imath</code>	∇	<code>\nabla</code>	\neg	<code>\neg</code>	\triangle	<code>\triangle</code>
j	<code>\jmath</code>	$\sqrt{}$	<code>\surd</code>	\flat	<code>\flat</code>	\clubsuit	<code>\clubsuit</code>
ℓ	<code>\ell</code>	∂	<code>\partial</code>	\natural	<code>\natural</code>	\diamondsuit	<code>\diamondsuit</code>
\wp	<code>\wp</code>	\top	<code>\top</code>	\sharp	<code>\sharp</code>	\heartsuit	<code>\heartsuit</code>
\Re	<code>\Re</code>	\bot	<code>\bot</code>	$\ $	<code>\ </code>	\spadesuit	<code>\spadesuit</code>
\Im	<code>\Im</code>	\vdash	<code>\vdash</code>	\angle	<code>\angle</code>	\Join	<code>\Join</code>
\mho	<code>\mho</code>	\dashv	<code>\dashv</code>	\backslash	<code>\backslash</code>	∞	<code>\infty</code>

4.15.5 具有两种尺寸的符号

有些符号具有两种尺寸, 位于行内公式时, 显示为小尺寸, 位于行间公式时, 显示为大尺寸, 下表列出了同一命令表示的不同大小的符号.

\sum	\sum	<code>\sum</code>	\cap	\cap	<code>\bigcap</code>	\odot	\odot	<code>\bigodot</code>
\int	\int	<code>\int</code>	\cup	\cup	<code>\bigcup</code>	\otimes	\otimes	<code>\bigotimes</code>
\oint	\oint	<code>\oint</code>	\sqcup	\sqcup	<code>\bigsqcup</code>	\oplus	\oplus	<code>\bigoplus</code>
\prod	\prod	<code>\prod</code>	\vee	\vee	<code>\bigvee</code>	\uplus	\uplus	<code>\biguplus</code>
\coprod	\coprod	<code>\coprod</code>	\wedge	\wedge	<code>\bigwedge</code>			

上述符号都可以使用上下标命令加上下限, 上下限的位置在行内公式和行间公式中可能有所不同. 总可以使用命令 `\limits` 强制上下限放在符号的上方和下方, 或者使用命令 `\nolimits` 强制上下限放在符号的侧旁. 参见第 47 页的例子.

4.15.6 函数名

在数学公式中显示变量名和函数名时有一个通行的标准, 就是用斜体显示变量名, 用正体显示函数名. 在 TeX 中通过在函数名前加上倒斜线, 将函数名与变量名区分开. 这样的函数名需要事先定义, L^AT_EX 已定义了下列函数名:

<code>\arccos</code>	<code>\cosh</code>	<code>\det</code>	<code>\inf</code>	<code>\limsup</code>	<code>\pr</code>	<code>\tan</code>
<code>\arcsin</code>	<code>\cot</code>	<code>\dim</code>	<code>\ker</code>	<code>\ln</code>	<code>\sec</code>	<code>\tanh</code>

<code>\arctan</code>	<code>\coth</code>	<code>\exp</code>	<code>\lg</code>	<code>\log</code>	<code>\sin</code>	<code>\bmod</code>
<code>\arg</code>	<code>\csc</code>	<code>\gcd</code>	<code>\lim</code>	<code>\max</code>	<code>\sinh</code>	<code>\pmod</code>
<code>\cos</code>	<code>\deg</code>	<code>\hom</code>	<code>\liminf</code>	<code>\min</code>	<code>\sup</code>	

最右列下方两个命令都生成函数 mod, 所以把它两个排在一起, 未按字母次序排在表中. 这两个函数用法不同, 输出形式也不同, 示例如下: 输入 `$a\bmod(m+n)b$`, 输出为 $a \bmod (m + n)$, 输入 `$a\pmod{m+n}$`, 输出为 $a \pmod{m + n}$.

第五章 常用的文档的类别与版式

在 \LaTeX 2 ϵ 源文件导言部分的头部有一条文档类别命令, 确定整篇文档的全局处理格式, 命令形如

`\documentclass[可选项]{文档类别}`

其中文档类别有 4 种:

<code>book</code>	书籍类
<code>report</code>	报告类
<code>article</code>	文章类
<code>letter</code>	书信类

必须从中选用一种而且只能选用一种. 不同的文档类别具有不同的默认版式和不同的成分, 例如 `book` 类, 允许含有 `chapter` (章), 并对奇偶页有不同的处理方式, 而 `article` 类就没有“章”这一层次. 一般我们书写数学论文往往选用“`article`”类.

学完本章, 读者就可以尝试打印自己的论文了. 如果在排版中发现问题或者想使自己论文的版面更漂亮些, 就需要参看“提高篇”了.

§5.1 文档类别命令中的可选项

文档类别命令中可选项的作用范围是整篇文档, 本章先介绍一些最常用的可选项. 注意所有选项都不能加前导符“ \backslash ”. 如果使用了多个互不排斥的选项, 这些选项应用逗号分开 (只能用西文逗号, 不可用中文逗号).

5.1.1 指定基本字体尺寸

指定基本字体尺寸的选项有 3 种:

`10pt` `11pt` `12pt`

只能选用一种. 若都不选用, 则默认是选用 10pt. 除非在文稿中加入了另外的声明, 否则所有普通文本都使用这个指定的字体尺寸, 其它变大变小的字体都是相对于这个尺寸的, 各种标题、角标、脚注等都会随着这个指定尺寸的改变而自动改变大小. 注意如果正文是中文, 则选项 10pt, 11pt 和 12pt 应分别与天元命令

```
\def\ChineseScale{1000}
\def\ChineseScale{1095}
\def\ChineseScale{1200}
```

相配合, 使得汉字的大小能与西文匹配.

5.1.2 指定纸张大小

如果不想直接指定页芯的大小(见第 11 页), 可改为指定纸张的大小, L^AT_EX 2_ε 会根据指定的字体尺寸和纸张大小自动确定页芯的大小. 指定纸张大小的选项有如下几种, 括号中的数字是纸张尺寸的说明, 不要输入到源文件中.

a4paper (297×210mm)	executivepaper (10.5×7.25in)
a5paper (210×148mm)	legalpaper (14×8.5in)
b5paper (250×176mm)	letterpaper (11×8.5in)

默认值是 letterpaper, 即美国信纸尺寸.

通常情况下排版输出使用纵向模式(“肖像画”模式), 即纸张格式中长的方向是排版结果的竖直方向. 如果想使用横向模式, 即将纸张格式中短的方向变为排版输出结果的竖直方向, 可使用如下的“风景画”模式选项:

landscape

注意打印时纸张的摆放方向并不改变.

5.1.3 其它一些选项

与标题页有关的选项是

notitlepage	titlepage
-------------	-----------

对于 book 和 report 类, 标题默认打印在单独一页. notitlepage 选项使标题与开头的文本排到同一页上.

对于 `article` 类, 默认标题与正文排在同一页. `titlepage` 选项可使标题单独成页.

下面又是一对有相反作用的选项

<code>draft</code>	<code>final</code>
--------------------	--------------------

使用选项 `draft` 后, 若排版结果出现超长的行, 则在该行的右端显示一个粗黑条, 提醒用户注意. 默认值是 `final`, 无论一行超出多少, 也不显示粗黑条. 写初稿时先使用选项 `draft`, 等一切调整妥当后再改为 `final`.

§5.2 章节

在 $\text{\LaTeX} 2_{\epsilon}$ 中有一些划分章节的命令, 其一般格式是

<code>\章节命令[短标题]{标题}</code>
<code>\章节命令*{标题}</code>

可用的章节命令按从大到小的层次有如下几种

<code>\part</code>	<code>\chapter</code>	<code>\section</code>	<code>\subsection</code>
<code>\subsubsection</code>	<code>\paragraph</code>	<code>\subparagraph</code>	

命令中的短标题用于显示在目录和 `headings` 页面版式的页眉中, 如果不选用短标题, 就认为它和标题是一样的.

对于每个不带 * 号的章节命令, 除 `\part` 外都有一个内部计数器, 计数器的名称与去掉倒斜线的章节命令相同. 每调用一次不带 * 号的章节命令, 对应计数器的值就加 1, 并同时 will 将低层次的计数器重置为 0.

除 `\part` 外, 不带 * 号的章节命令会自动对章节进行顺序编号 (这是 \LaTeX 特色之一), 并可自动形成目录表和页眉标题. 带有 * 号的章节命令则不会自动编号, 也不会把章节标题放到目录表和页眉中.

每个章节命令都有一个层次号, `\section` 层次号总是 1, 向下逐次递增, 直到 `\subparagraph` 层次号为 5. 对于 `book` 和 `report` 类, `\part` 层次号为 -1, `\chapter` 层次号为 0. 对于 `article` 类, `\part` 层次号为 0, 没有 `\chapter` 这一层次.

自动编号时只对除 `\part` 外的前 3 个层次进行编号, 对于 `book` 和 `report` 类, 就是对章 (`\chapter`)、节 (`\section`) 和小节 (`\subsection`) 进行编号, 即编号到层次号为 2 的层次. 对于 `article` 类, 则是对 `\section`、`\subsection` 和 `\subsubsection` 进行编号, 即编号到层次号为 3 的层次. 计数器 `secnumdepth` 用

于记录编号到达的最深层次数, 由上可知, 对于 `book` 和 `report` 类, 其值为 2, 对于 `article` 类, 其值为 3.

命令

```
\setcounter{secnumdepth}{数}
```

用于扩展或减小章节编号的层次. 对于 `book` 和 `report` 类, 数的取值范围是 -2 到 5; 对于 `article` 类, 数的取值范围是 -1 到 5. 细心的读者会发现数的范围中出现了不存在的层次数, 使用这个不存在的层次数意味着禁止所有的章节编号.

L^AT_EX 内部有一批计数器, 储存当前的章节号, 它们的名称就是相应的章节命令除去 ‘\’, 即: `part`, `chapter`, `section`, `subsection`, `subsubsection`, `paragraph`, `subparagraph`. 如果想改变章节计数器的初始值, 可用如下形式的命令

```
\setcounter{章节计数器的名称}{数}
```

这个命令应放在章节命令之前, 新的章节的序号值是这个初始值加 1.

下面再介绍在 `article` 类的默认格式下, 如何输入中文章节标题. 由于节标题的西文字体是 `\Large\bfseries`, 因此中文标题应添加天元命令 “\大\黑”; 小节标题的西文字体是 `\large\bfseries`, 因此中文标题应添加天元命令 “\中\黑”; 再往下的标题的西文字体都是 `\normalsize\bfseries`, 因此中文标题均应添加天元命令 “\黑”. 而 `\part` 标题比较特殊, 它被分成两行, 第一行输出 “{\Large\bfseries Part I}”, 第二行再输出标题, 用的字体则为 `\huge\bfseries`, 因此相应的中文标题应添加天元命令 “\双\黑”. 如果要修改第一行出现的名称 “Part”, 可用以下命令:

```
\renewcommand{\partname}{\大\黑新的名称}
```

例如可改为中文 “篇” 或 “章”, 但编号仍在后面, 变成 “篇 I”, 不合中文习惯. 所以这个层次的标题对中文文章不太合用, 如要自己定义修改, 则可参看本书的 “提高篇”. 下面的例 5-2-1.ty 展示了在 `article` 的默认格式下章节标题的样式以及它们与正文之间的留空情况.

```

% 5-2-1.ty
\def\ChineseScale{1000}
\input tyinput
\documentclass{article}
\begin{document}
\part{\双\黑篇(Part)的标题}
本篇将介绍基本概念与结论. \setcounter{section}{3}
\section{\大\黑节(Section)的标题}
本节包括以下内容.
\subsection{\中\黑小节(Subsection)的标题}
本小节将介绍以下基本概念与结论.
\subsubsection{\黑子节(Subsubsection)的标题}
本子节将介绍以下基本概念与结论.
\paragraph{\黑段(Paragraph)的标题}
本段将介绍以下基本概念与结论.
\subparagraph{\黑子段(Subparagraph)的标题}
本子段将介绍以下基本概念与结论.
\end{document}

```

其输出如下页所示.

§5.3 文章标题

在 \LaTeX 中定义了几个有关文章标题的命令:

```

\title{标题文本, 如含汉字, 则应添加命令'\特'}
\author{作者信息, 如含汉字, 则应添加命令'\中'}
\date{日期文本, 如含汉字, 则应添加命令'\中'}
\thanks{脚注文本, 如含汉字, 则应添加命令'\小'}
\maketitle

```

\LaTeX 用特定的字体字号居中打印标题内容, 为了使汉字的大小与西文匹配, 必须加入相应的天元字号命令, 如“\特”、“\中”、“\小”等等. 如果标题过长, 会自动分行. 作者也可用命令\\人工分行, 此时命令(以\title为例)形如

```
\title{...\\\...\\\...}
```

Part I

篇(Part)的标题

本篇将介绍基本概念与结论.

4 节(Section)的标题

本节包括以下内容.

4.1 小节(Subsection)的标题

本小节将介绍以下基本概念与结论.

4.1.1 子节(Subsubsection)的标题

本子节将介绍以下基本概念与结论.

段(Paragraph)的标题 本段将介绍以下基本概念与结论.

子段(Subparagraph)的标题 本子段将介绍以下基本概念与结论.

如果有多个作者, 其作者信息可以用 \and 相互隔开, 如

```
\author{作者1\\工作单位1\\地址1
\and 作者2\\工作单位2\\地址2}
```

排版后形如

作者1	作者2
工作单位1	工作单位2
地址1	地址2

如果不想左右并排而是上下排列, 只需用 \\[距离] 代替 \and.

如果没有给出 \date 命令, 系统会在作者下面自动加上当前日期. 如果给出了 \date 命令, 则不再打印当前日期, 而是在打印日期的地方打印日期文本, 但这个日期文本不一定是日期, 可以是任何文本. 命令 \date{} 相当于取消了 \date 的作用.

命令 \thanks 可以出现在 \title、\author 和 \date 命令参数中的任何地方. 它在出现的位置自动加上一个标记, 同时在标题页上以带有相同标记的脚注形式排版脚注文本.

上述命令仅仅给出了与标题有关的信息, 要想真正打印出这些信息, 必须使用命令 `\maketitle`, 该命令在它出现的位置开始新的一页打印标题页. 因为文章标题通常要打印在最前面, 所以这个命令通常是正文主体的第一条命令, 即紧接在 `\begin{document}` 之后.

对于 `book` 和 `report` 类, 标题页是单独一页, 只含有 `\title`、`\author` 和 `\date` 命令提供的标题信息. 对于 `article` 类, 标题页上除了标题信息外, 下面还有正文. 文档类别命令中的选项 `titlepage` 和 `notitlepage` 可以改变上述默认设置.

如果不想使用上述几个标题命令生成格式化的标题, 而想随心所欲排版标题页, 则应使用下述环境:

```
\begin{titlepage}
  标题页内容
\end{titlepage}
```

其中标题页内容可以含有排版命令和文本, 因此可以排成任何想要的形式. 这个环境结束时, 在新的一页显示排好了的标题页, 所以这个环境不需要 `\maketitle` 命令. 对于 `article` 类, 也可以利用这个环境打印单独的标题页.

注意上述环境总是重置页码计数器, 从而使标题页的页码是1 (但不显示这个页码), 后继页码从2开始. 为了保持页码的连续性, 上述环境最好是放在正文的开始部分.

§5.4 摘要

生成摘要的环境是

```
\begin{abstract}
  摘要文本, 如含汉字, 则应添加命令'\九'
\end{abstract}
```

在 `book` 类中没有这种摘要; 在 `report` 类中, 摘要位于单独一页并带有页码; 在 `article` 类, 摘要与文章标题打印在同一页, 但若选用了文档类选项 `titlepage`, 摘要也是单独占一页的.

不过摘要环境自动生成的标题是英文“Abstract”, 为了生成中文标题, 我们可在 使用此环境前插入以下命令:

```
\def\abstractname{\黑摘\quad 要}
```


下面的例子 5-4-1.ty 展示了论文首页的输入方法, 书写数学论文时可以照抄其中的控制命令.

```
% 5-4-1.ty
\def\ChineseScale{1200}
\input tyinput
\documentclass[a4paper,12pt]{article}
    % [...]为可选项
\usepackage[indentfirst,latexsym,bm]
\title{{\特广义特征值问题}$Ax=\lambda Bx$}
\thanks{Supported in part by NSFC.}
\footnotetext{\it{Keyword}: {\小广义特征值, ...}}
} % 标题、关键字等
\author{Zhang Aiguo {\中\仿(张爱国)}}\[5pt]
Department of Mathematics, ECNU\
{\中\仿(华东师范大学数学系)}
} % 作者、工作单位
\date{2001年4月1日}
    % {}内可写任意字符或为空白, 无此句则打印当前时间
\setlength{\parindent}{2em} % 设置段首缩进量
\renewcommand{\baselinestretch}{1.2} % 重设行距
\begin{document}
\maketitle % 打印 \title、\author、\date 等内容
\def\abstractname{\黑摘\quad 要}
\begin{abstract}
An abstract {\九摘要}
\end{abstract}
    ... 正文
\end{document}
```

相应的输出可参见另图.

广义特征值问题 $Ax = \lambda Bx^*$

Zhang Aiguo (张爱国)

Department of Mathematics, ECNU

(华东师范大学数学系)

2001年4月1日

摘 要

An abstract 摘要

... 正文

* *Keyword:* 广义特征值, ...

*Supported in part by NSFC.

第六章 自定义与改错

本章首先将介绍如何定义你自己的命令,使得命令变得更容易记忆,或者使它们更合你的心意.同时你还可以根据自己的需要改变各种计数器以及长度的值,从而在设计版面方面拥有更多自主权. \TeX 的特点就是用户拥有极大的自由度,它实际上是一种编程语言,你了解得越深入,能做的事越多.当然作为基本篇,我们只介绍一些最表层的用法.在本章的后半部分简要地介绍了 \TeX 的出错信息以及警告信息,告诉你遇到这类信息时的处理原则.本章也是基本篇的结束,希望读者在学完本章后能够多多实践,然后当感到有提高的必要时,再去有的放矢地阅读提高篇,最终成为 \TeX 的能手.

§6.1 自定义命令

\LaTeX 本身提供的定义命令的语句有以下两种:

<pre>\newcommand{新定义的命令名}[参量个数]{命令内容}</pre>
<pre>\renewcommand{重新定义的命令名}[参量个数]{命令内容}</pre>

这里参量个数不能超过9个.很显然,第一种语句是定义原来没有的命令,第二种语句则是对原来已有定义的命令重新定义.这样做当然出于系统安全的目的.因为很多时候你并不知道所选定的命令名是否已经定义过,如果在无意之中把已经定义过的命令重新定义,就有可能产生无法预料的后果,而你甚至不知道错误起因于何处.现在当你新定义一个已经有定义的命令时, \LaTeX 在编译时会有出错信息,提醒你改用别的名字.

命令定义语句的原理很简单:每当 \LaTeX 在编译时遇到一条命令,就会寻找这条命令的定义语句,然后把这条命令用命令内容来替代,如果带有参数,则把参数的值代入,替换完成后,再对新的语句作编译.因此命令语句本质上就是替换,至于复杂的命令所引起的递归嵌套等问题,就超出本书的范围了.

自定义命令的第一个用处是可以节省输入的工作量,或帮助记忆命令.例如为打印希腊字母 ϵ ,需要输入命令 “`\varepsilon`”,既长又容易出错.如果它在文

章中经常被用到, 那么可以给它取一个简称 “\eps” 或 “\e”, 即输入命令

```
\newcommand{\e}{\varepsilon}
```

自此以后, L^AT_EX 一看到 “\e” 就会自动把它替换成 “\varepsilon”. 不过为了单独输出 ε , 还是必须输入 $\$ \backslash e \$$. 为了省掉这两个美元号, 我们可以把定义改为

```
\newcommand{\e}{\$ \varepsilon \$}
```

但又会产生新的问题: 当 ε 出现在数学模式, 例如 $|x| < \varepsilon$ 中时, 如果输入 $\$ |x| < \backslash e \$$, 经替换后变成 $\$ |x| < \$ \backslash \varepsilon \$ \$$, 使 $\backslash \varepsilon$ 出现在数学模式以外, L^AT_EX 就会产生出错信息. 为了使这个命令不论在数学模式以内还是以外都能使用, 可利用 L^AT_EX 2_ε 的命令 $\backslash ensuremath$, 用法如下所示:

```
\newcommand{\e}{\ensuremath{\varepsilon}}
```

又如 L^AT_EX 的命令都是来源于英语, 这对于精通英语的人来说当然能帮助记忆, 但对于英语不太好的人就会感到难记, 只要拼错一个字母, 就会出错通不过编译, 令人恼火. 现在就可以自己给它换一个易记的名字. 例如把 “\triangle” 改名为 “\SJX” (源自 “三角形” 的汉语拼音), 既短又好记. 为此只要输入:

```
\newcommand{\SJX}{\ensuremath{\triangle}}
```

这里都采用大写字母是为了避免重名, 因为 L^AT_EX 的原始命令大都是小写字母.

复杂一点的命令还带有参量, 例如以下命令:

```
\newcommand{\anvec}[2]{\ensuremath{\#1_1, \ldots, \#1_{\#2}}}
```

方括号中的数字 2 表明新命令 “\anvec” 带有两个参量. 它们在后面用花括号括起来的命令内容中分别表示为 “#1” 与 “#2”.

在具体使用命令 $\backslash anvec$ 时, 必须把这两个参量分别用花括号括起来, 紧跟在命令 $\backslash anvec$ 的后面. 所谓 “紧跟”, 是指它们之间除了空格以及至多一个换行外, 没有别的字母插入. 此外当参量只含一个字母的话, 也可把花括号省去. 例如下面 $\backslash anvec\{u\}\{n\}$ 与 $\backslash anvec\ u\ n$ 有相同效果, 经替换后都相当于 $\$ u_1, \ldots, u_n \$$, 最后产生输出 u_1, \dots, u_n . 但是如果我们输入 $\backslash anvec\{u\}\{n+1\}$, 经替换后变成 $\$ u_1, \ldots, u_{n+1} \$$, 输出是 u_1, \dots, u_{n+1} , 而不是我们所期望的 $u_1, \dots, u_n + 1$. 问题显然出在第二个参量 $n+1$ 表面上看来是放在花括号内, 而实际上在替换时要去掉一层花括号, 就变成没有括号了. 为了避免这种差错, 我们可以在使用命令时多加一层括号, 输入为 $\backslash anvec\{u\}\{\{n+1\}\}$, 或者在定义时多加一重括号, 如下所示:

```
\newcommand{\anvec}[2]{\ensuremath{\#1_1, \ldots, \#1_{\{\#2\}}}}
```

这样就万无一失了. 希望大家重视这个问题, 一个最保险的办法就是给命令内容中出现的参量一律加上括号, 变成 $\{\#1\}, \dots, \{\#9\}$.

最后说一下自定义命令的作用范围. 如果定义语句出现在导言部分, 那么这个命令对整个文件都有效, 你在这个定义语句出现以后的任何地方修改定义都必须使用 `\renewcommand`. 而在文件的正文中, 各种环境以及由 `{}` 构成的集团又使得文件内部被分成许多层次. 外层的命令在内层继续有效, 而内层定义的命令的作用范围是局部的, 一旦退出这个层次, 这个命令就不起作用了. 因此在内层修改外层的命令时必须用 `\renewcommand`, 但一旦退出这个层次, 这个命令又自动恢复原先的意义. 一个在内层被定义过的命令如果在它的作用范围之外再次被定义, 则仍需使用命令 `\newcommand`, 这应该是很自然的. 因此在定义新的命令时要注意它的作用范围, 再决定用作局部定义还是整体定义. 一般我们可以把自己常用的定义统一放在文件的前言部分, 以利维护.

§6.2 给计数器和长度赋值

L^AT_EX 内部有不少计数器, 我们已经接触到的有以下一些:

<code>part</code>	<code>subsection</code>	<code>subparagraph</code>	<code>footnote</code>
<code>chapter</code>	<code>subsubsection</code>	<code>page</code>	
<code>section</code>	<code>paragraph</code>	<code>equation</code>	

它们的意义从字面上就能看出, 例如 `page` 计数器就存放着当前的页码. 如果我们h需要改变计数器的值, 则可使用下列命令:

```
\setcounter{计数器名称}{数字}
\addtocounter{计数器名称}{数字}
\stepcounter{计数器名称}{数字}
```

这里的数字都应是整数, 其中 `\setcounter` 是给计数器赋值, `\addtocounter` 是把指定的值加到计数器上, `\stepcounter` 则把指定计数器的值加1, 同时使下属计数器的值置零. 例如执行 `\stepcounter{section}` 后, `section` 计数器的值增加了1, 同时它下面的计数器 `subsection`, `subsubsection`, `paragraph`, `subparagraph` 都被置零.

如果你在文章中想打印某个计数器的当前值, 可使用命令

```
\the计数器名称
```

例如输入“本页是第 `\thepage` 页”就能把页码打印出来.

为了改变某些长度参数, 可以使用以下命令:


```
\setlength{长度命令}{长度值}
\addtolength{长度命令}{长度值}
```

例如我们已经遇到过的 `\setlength{\parindent}{2em}`。

这里再介绍两个有用的长度：正文宽度 `\textwidth` 以及正文净高度（不包括页码和页眉）`\textheight`。这样我们遇到过的长度命令有

<code>\arraycolsep</code>	<code>\medskipamount</code>	<code>\smallskipamount</code>
<code>\baselineskip</code>	<code>\parindent</code>	<code>\textwidth</code>
<code>\bigskipamount</code>	<code>\parskip</code>	<code>\textheight</code>

长度值应该是带有单位的长度，例如 `3mm` 或者别的已有定义的长度的倍数，例如 `0.5\textwidth`，`-\smallskipamount` 等。

§6.3 出错信息

\LaTeX 的出错信息有两个来源，一部分来自 \LaTeX 本身，还有一部分来源于底层的 \TeX 。一个 \LaTeX 错误可能会带来一系列 \TeX 的错误信息，这是因为 \LaTeX 的操作是位于 \TeX 之上的。

下面我们先看一个简单的例子 (`6-3-1.tex`):

```
% 6-3-1.ty
\documentclass{article}
\begin{document}
The last words appear in \txetbf{bold face}.
\end{document}
```

这里有一个错：`\textbf` 被误输入成 `\txetbf`。用 \LaTeX 编译时，屏幕上会显示以下出错信息：

```
! Undefined control sequence.
1.4 The last words appear in \txetbf
                                {bold face}.
?
```

这个出错信息是 \TeX 发出的，因为 \LaTeX 发出的出错信息的头部是

! LaTeX Error:

当然对大部分用户来说没有必要去区分它们. 这个出错信息告诉我们它遇到了“未被定义的控制序列”, 第2行的1.4指明错误出在源文件的第4行, 而且它刚刚把“\txetbf”读进去, 接在下一行的“{bold face}.”则是它将要读入的内容. 第4行的问号则是等待我们的处理意见.

如果我们再输入一个问号, 屏幕显示下面一段话:

```
Type <return> to proceed, S to scroll future error
messages,
R to run without stopping, Q to run quietly,
I to insert something, E to edit your file,
1 or ... or 9 to ignore the next 1 to 9 tokens of input,
H for help, X to quit.
?
```

我们的答复有以下几种选择:

1. 按“回车”键, 让 TeX 按预定的程序去处理各类错误, 同时继续往下编译;
2. 键入“S”再回车, 进入滚动模式(scroll mode), TeX 将不再停顿, 一直往下编译, 同时不断显示遇到的出错信息, 直到最终停止. 这相当于对所有的出错信息都按回车键;
3. 键入“R”再回车, 进入运行模式(run mode), 这种模式与滚动模式类似, 但级别更高, 即在滚动模式会中断的地方(例如在遇到输入文件的命令, 但没有提供文件名时)它也不停止;
4. 键入“Q”再回车, 进入安静模式(quiet mode), 这种模式同 R 模式, 只是不在屏幕上显示任何出错信息;
5. 键入“I”再用键盘输入你想插入的字符串, 例如遇到上述例子的错误, 可以输入“I\textbf”再回车, 就能继续编译, 并得到正确的结果. 不过请注意, 源文件并没有改动, 如果不修改源文件, 下次编译时仍会在这里出错. 如果你输入“I\stop”再回车, 则 TeX 会停止编译, 并输出到当前页为止的 dvi 文件;
6. 键入一个小于 100 的正整数再回车, 则会删除接下去要输入的这么多个符号(控制字或控制符算作 1 个符号), 并停下来等待你的进一步指示;

7. 键入“H”再回车, 这时屏幕会显示一些帮助信息, 提示你错误的可能原因及处理方法. 不过千万别寄托太大的希望, 经验证明, TeX 的帮助信息往往解决不了什么问题;
8. 键入“X”再回车, TeX 停止编译, 而且当前页不被输出到 dvi 文件;
9. 键入“E”再回车, 效果同“X”, TeX 停止编译, 如果 TeX 的编译程序与某个文字处理程序有关联的话, 就会自动进入文字处理程序, 并把光标移到出错的那一行.

注意上面输入的字母不分大小写, 都有相同效果.

LaTeX 的出错信息可以用同样的方式处理. 所有的出错信息都记录在与源文件同名, 但扩展名是 log 的文件上, 因此你不用为面对屏幕上飞滚而过的出错信息发愁, 只要打开 log 文件, 就可以一条条浏览出错信息并作适当处理. 我们的经验是遇到出错就按“Q”, 任其继续, 等停止后再根据 log 文件的纪录进行修改. 一般比较多的错误是输入打字错误, 还有常犯的错误是漏掉进入数学模式的美元号 \$ 或者花括号 {} 不配对. 有时 TeX 提示的错误原因不一定是真正的原因, 而仅是诱发的错误, 因此初学者要仔细审读原文, 查出错误的真正原因, 等以后经验丰富了, 查错也变得容易了.

如果编译的是含有汉字的天元源文件, 那么要修改的应该是 ty 文件, 不是 tex 文件. 因为 tex 文件是不断更新的中间文件, 只有天元源文件才是要保存的. 当源文件里不含 Tydraw 的作图命令时, 天元不会影响行号, 因此出错信息的行号可以用在 ty 源文件里. 但是如果源文件里含有 Tydraw 的作图命令, 经天元处理后行数会增加, 行号的对应关系被破坏, 使得查错的难度也增加了. 如果错误隐藏得深难以查找的话, 不妨先把 Tydraw 的作图命令移去, 等编译通过后再插入.

§6.4 警告信息

这里只介绍两类最重要的警告信息: Overfull 以及 Underfull.

如果在水平方向超长, TeX 就会发出如下警告信息:

```
Overfull \hbox (10.00104pt too wide) in paragraph at lines 5--6
```

告诉你源文件第 5 至 6 行的地方超长了 10.00104pt. 如果是 draft 模式, 那么在 dvi 文件的屏幕显示或打印时会在超长的地方出现一个黑方块, 以引起注意. 对于西文的原稿, 可用插入隐含连字符“\”的方法帮助 TeX 换行, 以避免超长, 也可以人工干预, 用 \linebreak 强迫换行的方法解决问题. 如果超长不多, 也可不予处理, 在定稿时改用 final 模式, 使得黑方块消失, 一般不注意是看不出来的.

如果在竖直方向超长, TeX 会发出如下警告信息:

```
Overfull \vbox ...
```

告诉你页面的下端超长了. 如果超得不多, 一般可不予处理, 如果超得多了, 就要采取调整版面内容或强迫换页的方式来解决.

如果看到以下警告:

```
Underfull \hbox (badness 5504) in paragraph at lines 5--7
```

就是提醒你这里发生了太空松的状况, 其严重程度是5504. 严重程度以10000封顶, 如果达到10000就是非常严重了. 空松往往是由不适当的强迫换行引起的, 一般需要人工干预加以调整. 如果严重度不大, 也可以不处理.

如果在竖直方向出现空松, 会出现如下警告:

```
Underfull \vbox ...
```

读者同样可根据其严重程度决定是否需要处理.

提高篇

第七章 图形

L^AT_EX 提供了若干绘图命令,可以用来绘制直线、矢量线、圆、矩形、圆角矩形(卵形线)、Bézier 曲线等基本图形. 在 L^AT_EX 源文件中还可以插入一些外部图形. 用 L^AT_EX 命令画图时,对线段的倾角和圆的直径都有一定的限制,如果使用天元内嵌的绘图功能,则可画任意斜率的线段和任意半径的圆,并且还有更强的绘图功能,例如只要给出曲线的方程,就能画出二维或三维曲线的图形.

§7.1 图形与坐标系

精确绘图离不开坐标系,在 L^AT_EX 中使用的是右手直角坐标系,水平轴为 x 轴,向右为正,竖直轴为 y 轴,向上为正. 在两个坐标轴上设置同样大小的单位长度 (`\unitlength`),习惯上令单位长度为 1mm,用其他的值也可以,例如令单位长度为 0.8mm. 图形中的所有长度都是以单位长度作为度量单位. 默认值是 1pt,重置单位长度的命令是

```
\setlength{\unitlength}{长度}
```

当一个图画好后,只要重设单位长度,就相当于对图形作放大或缩小变换.

绘制的任何一个图形总是和一个坐标系固连在一起,坐标轴的原点可以放在任何地方,通常是放在图形区域的左下角.

在 L^AT_EX 中绘制图形需要使用绘图环境:

```
\begin{picture}(宽度,高度)(x坐标,y坐标)  
    各种绘图命令  
\end{picture}
```

环境中的(宽度,高度)是图形占据的区域的宽度和高度,更确切地说,是 L^AT_EX 为插入图形所保留的区域的宽度和高度,换言之,如果实际绘制的图形大于这个尺寸,就可能与其它文本重叠,如果实际图形远远小于这个区域,排版出来就会出现很大的空白,所以应尽量使这个宽度和高度是实际图形的大小.

TeX 把一切对象都看成是盒子, 图形也是一个盒子, 这个盒子的宽度和高度就是绘图环境中指定的(宽度, 高度), 盒子的深度是 0, 盒子的基准点(参考点)就是盒子的左下角, LaTeX 处理这个图形盒子就像是处理一个的特大的字符盒子.

绘制的任何图形都固连着一个坐标系. 绘图环境中的(x 坐标, y 坐标)是图形盒子的基准点在这个坐标系中的坐标, 实质上是确定了坐标系在盒子中的位置, 也就是确定了整个图形在盒子中的位置. 如果在绘图环境中省略了这个坐标, 则默认这个坐标为 $(0, 0)$. 当图形绘制完成后, 如果改变了这个坐标, 就相当于将坐标系在盒子中作了平移. 因为坐标系是固连在图形上的, 所以也就是将图形在图形盒子中作了平移.

确定图形元素在坐标系中位置的命令(定位命令)有两个:

```
\put(x, y){图形元素}
\multiput(x, y)(\Delta x, \Delta y){个数}{图形元素}
```

其中图形元素可以是字符串或是下一节中要讲的能够产生图形元素的绘图命令. (x, y) 是图形元素基准点的坐标. 命令 `\multiput` 产生指定个数的图形元素, 这是通过把同样的图形元素进行平移产生的, 平移向量是 $(\Delta x, \Delta y)$. 例如

```
\multiput(3.0, 4.0)(1.2, 2.5){3}{图形元素}
```

就是在 3 个位置 $(3.0, 4.0)$, $(4.2, 6.5)$, $(5.4, 9.0)$ 画同样的图形元素.

§7.2 基本绘图命令

7.2.1 直线和矢量线

使用 LaTeX 的绘图环境可以绘制任意长度的水平直线段(横线)、竖直直线段(竖线)和某些特定斜率的斜线段, 命令为

```
\line(\Delta x, \Delta y){长度}
```

对于横线和竖线, 长度是它们的实际长度(以 `\unitlength` 的值作单位), 对于其他倾斜的直线段, 这个长度是它们在横轴上的投影的长度. 直线段的起点由定位命令 `\put` 或 `\multiput` 给出的坐标 (x, y) 所确定, 直线的方向数是 $(\Delta x, \Delta y)$. 没学过直线方向数的读者可如下理解方向数: 从直线段上的起点开始, 沿 x 轴方向移动 Δx 距离, 再沿 y 轴方向移动 Δy 距离, 就又回到了直线段或它的延长线上. 对方向数有如下限制:

- 必须是整数;

- 取值只能是 $0, \pm 1, \pm 2, \pm 3, \pm 4, \pm 5, \pm 6$;
- Δx 与 Δy 不能有公因子.

对于斜线, 长度的值不能小于 10pt 或 3.5mm, 否则不会显示出来, 对于横线和竖线无此限制. 为方便, 以下将直线段称之为直线.

绘制矢量线 (即带有箭头的直线) 命令是:

`\vector(Δx , Δy){长度}`

参数的含义与 `\line` 命令的参数一样, 与直线的区别只是在终点画上了箭头.

对于矢量线的方向数有更严格的限制, 除了必须是整数且无公因子外, 取值只能是 $0, \pm 1, \pm 2, \pm 3, \pm 4$;

当矢量线不是水平或竖直时, 对长度的值的限制与直线一样, 不能小于 10pt 或 3.5mm, 否则显示出来只有箭头部分而没有直线部分.

指定直线或曲线粗细的声明是细线声明或粗线声明:

`\thinlines` 或 `\thicklines`

默认是细线声明 `\thinlines`. 声明的作用直到遇到相反的声明为止, 或遇到所在环境的结束.

对于横线和竖线, 包括水平和竖直的矢量线的直线部分以及用横线和竖线构成的方框, 可用下列声明指定线的粗细:

`\linethickness{粗细}`

参数值粗细是正的长度, 必须带有单位, 例如 `{2pt}`, 它与 `\unitlength` (单位长度) 无关. 这个声明遇到细线声明或粗线声明之后就失去作用, 且只对横线和竖线起作用. 对于水平或竖直的矢量线, 当直线部分过粗时, 将会遮住箭头.

下面是一个例子 (见 7-2-1.tex), 输入

```
\begin{center}
\setlength{\unitlength}{1mm}
\begin{picture}(60,30)
\linethickness{1pt}
\put(0,0){\vector(1,0){60}}
\put(0,0){\vector(0,1){30}}
\thicklines
\put(5,7){\line(5,2){50}}
```

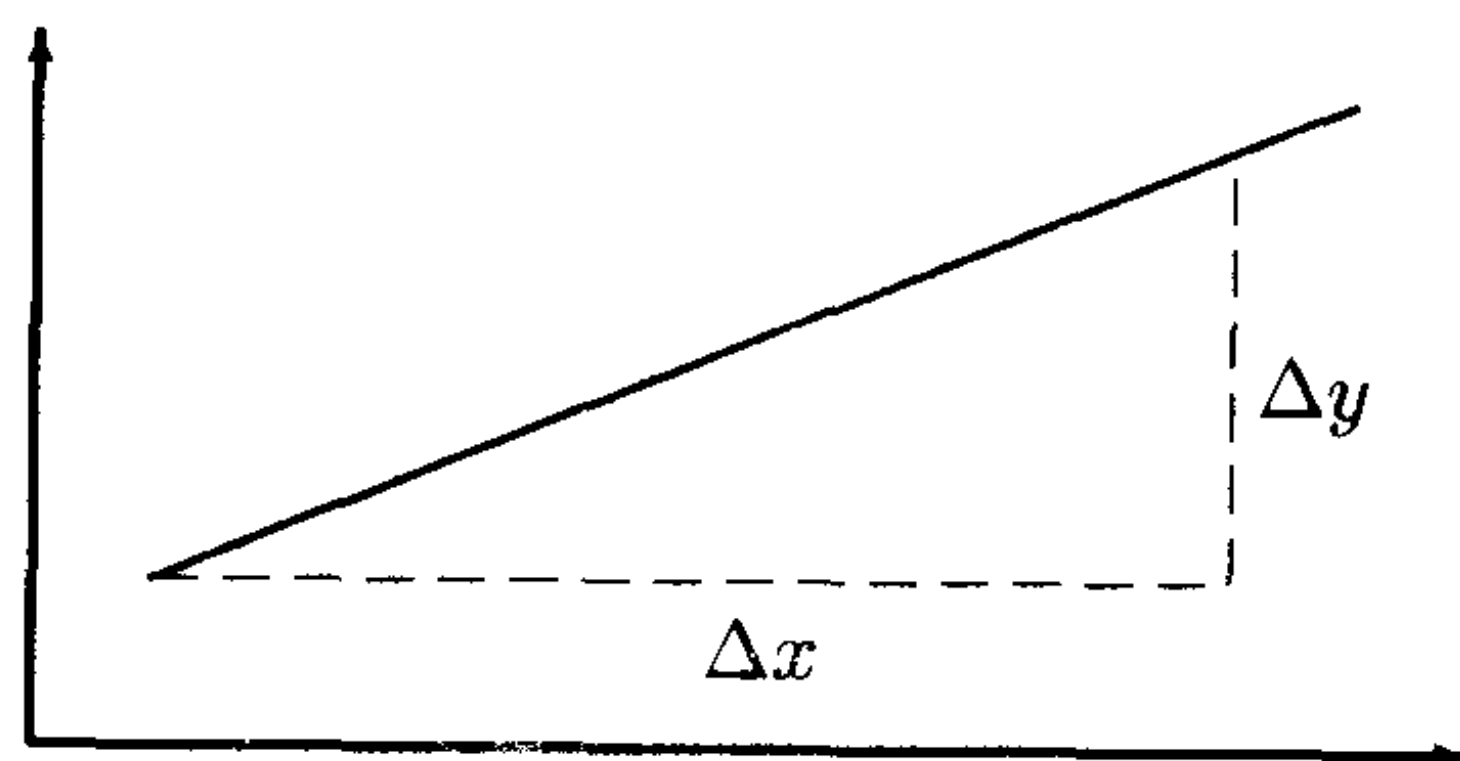


```

\thinlines
\multiput(5,7)(3,0){15}{\line(1,0){2}}
\multiput(50,7.00)(0,3){6}{\line(0,1){2}}
\put(28,3){$\Delta x$}
\put(51,14){$\Delta y$}
\end{picture}
\end{center}

```

输出为



除非特别说明,以后所有例子均使用 1mm 作单位长度.

7.2.2 圆和圆角矩形

画圆的命令是

```

\circle{直径}
\circle*{直径}

```

其中第一个命令只画出圆周,第二个命令(带*号的命令)画实心圆. 圆周是由若干段小圆弧组成. 由于事先设计好的小圆弧只有有限多种,并不具有任意曲率,所以只能画特定尺寸的圆, L^AT_EX 会选出与指定直径最接近的圆. 在 L^AT_EX 中圆的直径不能超过 40pt (约 14mm), 实心圆直径不能超过 15pt (约 5.3mm). 有些软件不用小圆弧拼成圆,就没有上述限制了.

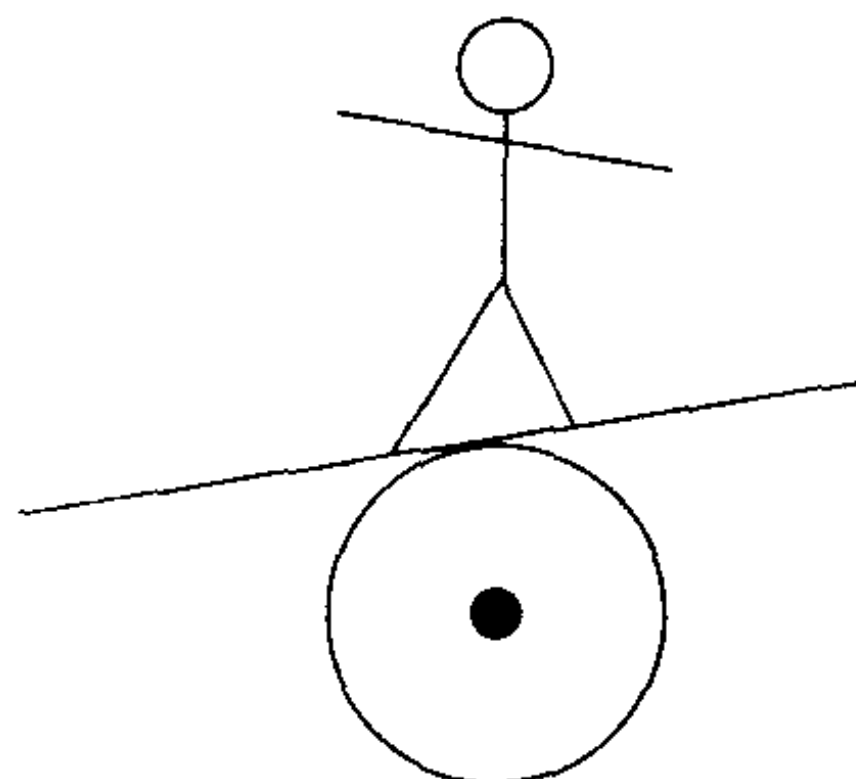
画圆时定位命令 `\put` 或 `\multiput` 中的坐标是圆心的坐标.

下面是一个例子(见 7-2-2.tex), 只用到了圆和直线:

```

\setlength{\unitlength}{1mm}
\begin{center}
\begin{picture}(35,32)
\put(20,7){\circle*{2}}
\put(20,7){\circle{14}}
\put(0.00,11){\line(6,1){35}}
\put(15.5,13.6){\line(3,5){4.5}}
\put(20,21){\line(1,-2){3}}
\put(20,21){\line(0,1){7}}
\put(13,28){\line(6,-1){14}}
\put(20,30){\circle{4}}
\end{picture}
\end{center}

```



所谓圆角矩形就是一个矩形的四角被四分之一圆代替. \LaTeX 对圆的直径有一定限制, 圆角矩形四角上圆的直径在不超过矩形短边长度的条件下, 总是尽量的大. 这种图形也称为卵形线. 除了完整的圆角矩形, 二分之一或四分之一的圆角矩形也经常用到, 对应的命令是

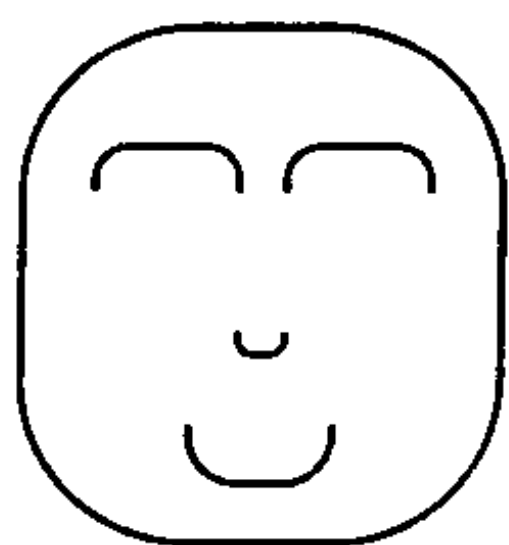
`\oval(宽度, 高度)[部分]`

其中的(宽度, 高度)是圆角矩形的外切矩形宽度和高度. 定位命令 `\put` 或 `\multiput` 中的坐标是整个圆角矩形对称中心的坐标. 可选项参数部分可以取如下值:

- t top, 上半部圆角矩形.
- b bottom, 下半部圆角矩形.
- l left, 左半部圆角矩形.
- r right, 右半部圆角矩形.
- tl 左上角四分之一圆角矩形.
- tr 右上角四分之一圆角矩形.
- bl 左下角四分之一圆角矩形.
- br 右下角四分之一圆角矩形.

对于圆来说, 没有画部分圆周的参数. 通过把圆角矩形的高和宽取成相同的值, 可以得到半个或四分之一圆周. 与圆的限制一样, 不能得到任意直径的部分圆周.

下面是用圆角矩形曲线画的一个图(见 7-2-3.tex).



```

\begin{picture}(20,22)
\thicklines
\put(10,11){\oval(20,22)}
\multiput(6,15)(8,0){2}{\oval(6,4)[t]}
\put(10,9){\oval(2,2)[b]}
\put(10,5){\oval(6,5)[b]}
\end{picture}

```

7.2.3 图形中的盒子

在绘图环境中, 将无框盒子和有框盒子的命令作了推广, 并增加了虚线框盒子, 它们是:

```

\makebox(宽,高)[位置]{文本}
\framebox(宽,高)[位置]{文本}
\dashbox{虚线长度}(宽,高)[位置]{文本}

```

其中(宽,高)指定矩形盒子的宽度和高度, 均以`\unitlength`的值为单位. 可选项参数位置指定文本在盒子中的位置, 可取如下值:

- t top, 文本水平方向居中, 竖直方向靠在盒子顶部.
- b bottom, 文本水平方向居中, 竖直方向靠在盒子底部.
- l left, 文本竖直方向居中, 水平方向靠在盒子左边.
- r right, 文本竖直方向居中, 水平方向靠在盒子右边.
- s stretch, 文本竖直方向居中, 水平方向伸展充满盒子.
- tl top left, 文本位于盒子的左上角.
- tr top right, 文本位于盒子的右上角.
- bl bottom left, 文本位于盒子的左下角.
- br bottom right, 文本位于盒子的右下角.

如果省略了可选项参数位置, 则文本在水平方向和竖直方向上都是居中地放置在盒子中. 可选项中有两个字母时, 顺序无关紧要, 例如[tl]与[lt]的作用相同.

命令中的虚线长度是组成虚线的小短线的长度, 两个小段线之间的空白也是这个长度, 以`\unitlength`的值为单位. 当盒子的宽度和长度是短线长度的整数倍时, 虚线框要好看一些.

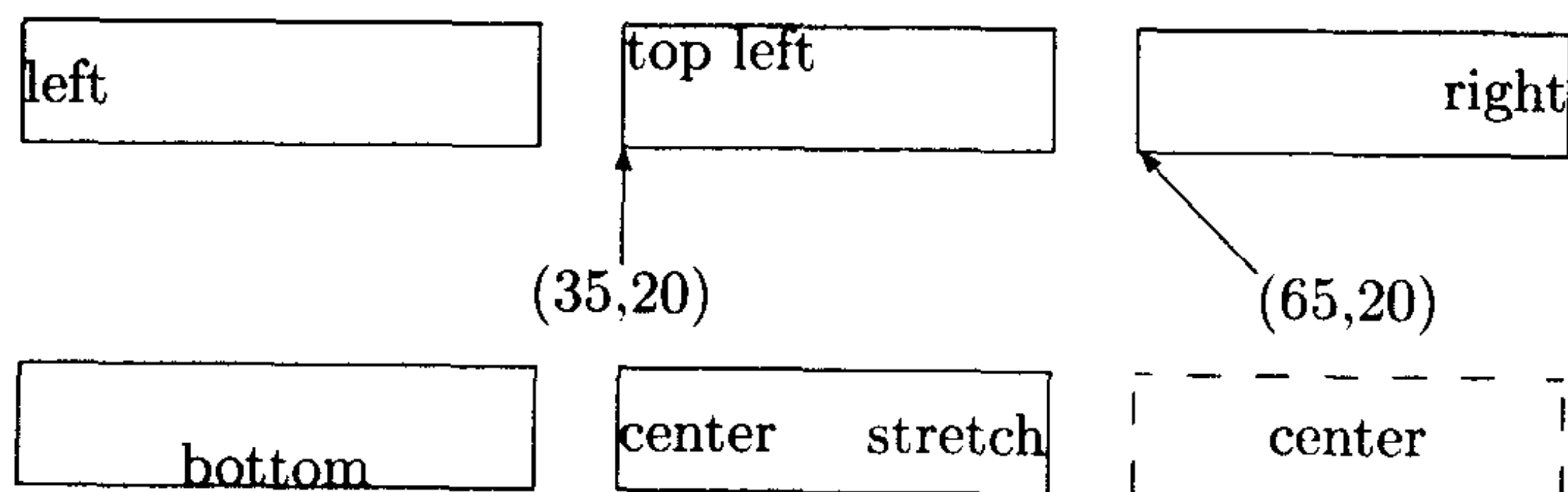
将盒子命令作为图形元素放在`\put`或`\multiput`命令中时, 定位坐标指定了盒子基准点的位置, 也就是盒子左下角的位置. 下面是一个例子(见7-2-4.tex), 输入

```

\setlength{\unitlength}{1mm}
\begin{picture}(90,27)
\put(0,0){\framebox(30,7)[b]{bottom}}
\put(35,0){\framebox(25,7)[s]{center\hfill stretch}}
\put(65,0){\dashbox{2}(25,7){center}}
\put(0,20){\framebox(30,7)[l]{left}}
\put(35,20){\framebox(25,7)[lt]{top left}}
\put(65,20){\framebox(25,7)[r]{right}}
\put(35,13){\makebox(0,0)[t]{(35,20)}}
\put(35,13){\vector(0,1){7}}
\put(72,13){\makebox(0,0)[t1]{(65,20)}}
\put(72,13){\vector(-1,1){7}}
\end{picture}

```

输出为



图中的箭头及坐标指出了两个盒子的基准点坐标.

图形元素 `\makebox` 是个没框的盒子, 使用最多的情况是把它宽度和高度都取为零, 使盒子变为一个点. 利用位置参数很容易确定文本与该点的相对关系, 从而容易把文本放置在任何所希望的位置上. 图中两个无框盒子都是一个点, 矢量线的起点与盒子的基准点坐标相同, 由此可直观看出文本与基准点的位置关系.

为了使图形与上下相邻文本有较大的竖直间隔, 应在绘图环境的前后插入竖直间距命令, 例如插入 `\medskip`.

仔细观察图中的数据, 可以算出所有图形元素占据的范围就是绘图环境给出的尺寸 (90, 27). 如果把这个尺寸改为 (90, 37), 并把图形区域左下角的坐标由默认值 (0, 0) 改为 (0, -5), 就会使图形区域在高度方向增加 10mm 的空白, 而且空白被分在了图形的上方和下方, 个中原因不再解释了, 请读者自行分析.

7.2.4 图形中的文本

在图形中插入文本, 最简单的方法是在定位命令后面直接写上文本:

```
\put(x,y){文本}
```

这样的文本不能太长, 因为它不会被换行. 坐标(x,y)是文本行基准点的位置, 粗略地说就是文本所占区域左下角的位置.

为了容易把文本放置在指定位置, 更常用的方法是使用上一节介绍的长宽为零的无框盒子:

```
\put(x,y){\makebox(0,0)[位置]{文本}}
```

在文字模式或绘图环境中都可排版竖直堆叠文本, 命令为

```
\shortstack[位置]{一系列文本}
```

其中位置可取值是 l, r 或 c, 默认是 c, 表示一系列文本在其所占区域中的对齐方式. 所占区域的左下角是基准点, 定位命令 \put 或 \multiput 指定的坐标就是该基准点的坐标. 竖直堆叠文本行之间的间隔很小, 显得紧凑. 下面是一个例子(见 7-2-5.ty), 输入

```
\begin{center}
\begin{picture}(63,15)
\put(0,0){\shortstack[l]{这是\\左对齐\\堆叠文本}}
\put(20,0){\shortstack[r]{这是\\右对齐\\堆叠文本}}}
\put(40,0){\shortstack{这是\\居中对齐\\堆叠文本}}
\put(60,0){\shortstack{A\\B\\C\\D}}
\end{picture}
\end{center}
```

输出为

这是	这是	这是	A
左对齐	右对齐	居中对齐	B
堆叠文本	堆叠文本	堆叠文本	C
			D

7.2.5 曲线

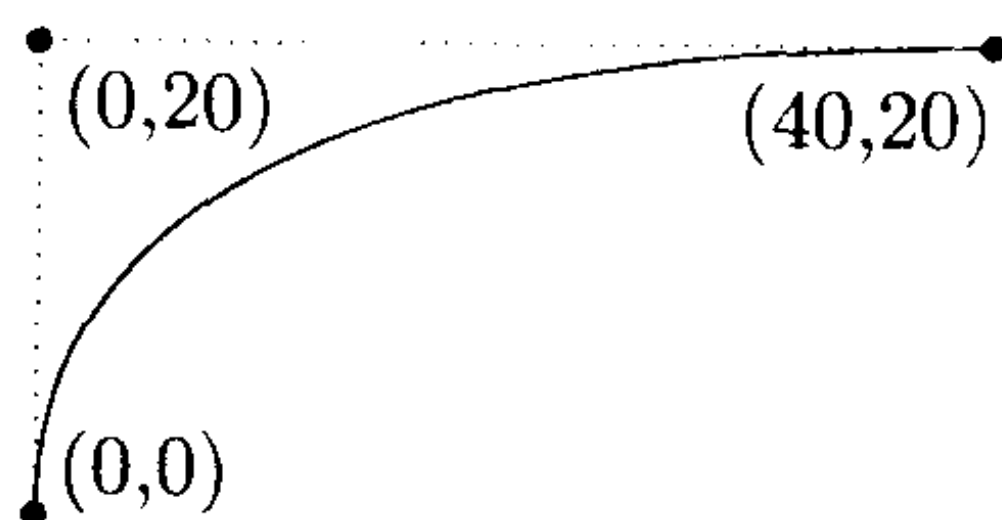
在 L^AT_EX 2_ε 中绘制曲线的命令是


```
\bezier{数}(x_1, y_1)(x_2, y_2)(x_3, y_3)
\qbezier[数](x_1, y_1)(x_2, y_2)(x_3, y_3)
```

它们都是画一条从 (x_1, y_1) 到 (x_3, y_3) 的 Bézier 曲线, 是用 数+1 个点画出来的. (x_2, y_2) 是控制曲线弯曲程度的控制点: 从该点到两个端点的连线是曲线的两条切线. 上面两条命令的区别是第二个命令中的画图点数为可选项, 如果省略该项, $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X} 2_{\epsilon}$ 会自动计算出合适的点数以画成一条光滑的 Bézier 曲线. 保留第一条命令仅仅是为了和以前的 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X} 2.09$ 版本兼容. 下面的例子直观显示了 3 个点的关系 (见 7-2-6.tex), 输入.

```
\begin{center}\begin{picture}(41,21)(-0.5,-0.5)
\put(0,0){\circle*{1}}    \put(0,20){\circle*{1}}
\put(40,20){\circle*{1}}  \qbezier[20](0,0)(0,10)(0,20)
\qbezier[40](0,20)(20,20)(40,20)
\qbezier(0,0)(0,20)(40,20)
\put(1,0){\makebox(0,0)[bl]{(0,0)}}
\put(1,19){\makebox(0,0)[t1]{(0,20)}}
\put(40,18.5){\makebox(0,0)[tr]{(40,20)}}
\end{picture}\end{center}
```

输出为



图中的虚线是用 Bézier 曲线画成的, 这只要使命令中的 3 个点位于一条直线上, 并且指定较少的点数即可. 利用这种方法也可以画出具有任意斜率的直线, 但要指定足够多的点数 (或不指定点数).

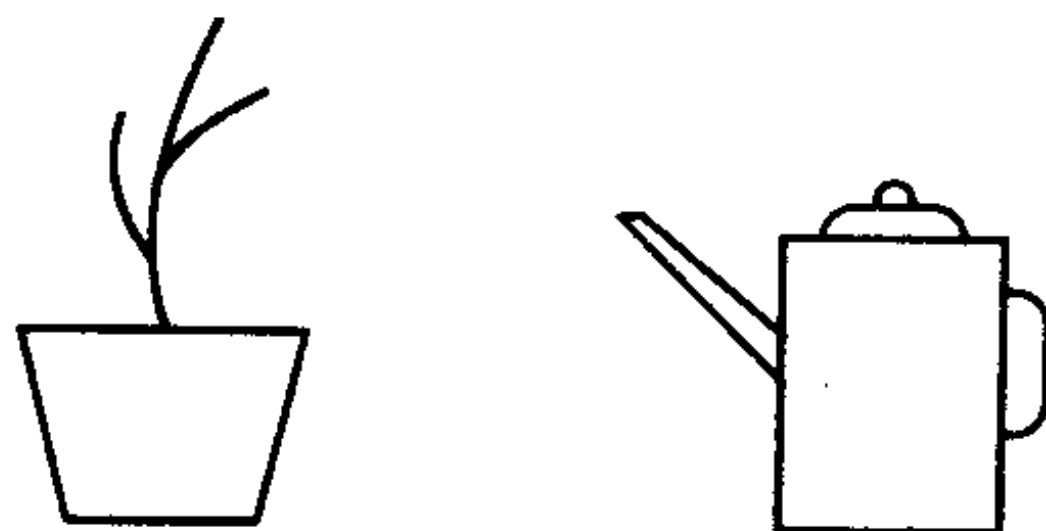
§7.3 子图

把图形中的部分图形当成一个整体处理, 这部分图形就称之为子图. 子图可以由图形环境产生, 称为子图环境. 这实际构成绘图环境的嵌套, 即在定位命令 `\put` 或 `\multiput` 中的图形元素是另一个绘图环境, 以生成子图. 子图环境有自

己的坐标系, 其定位命令中的坐标都是相对于该坐标系的. 在子图环境中可以设置自己的单位长度以及线的粗细, 若不设定, 则使用外部绘图环境的值. 当一个图形中含有几个图形时, 使用子图环境可以减少定位差错, 并且可以容易地调整改变相对位置. 下面是一个含有两个子图环境的例子, 源文件缩行输入, 便于看出每个子图环境的语句, 容易检查错误. 输入是(见 7-3-1.tex):

```
\begin{center}
\begin{picture}(44,21)
\thicklines
\put(0,0){\begin{picture}(12,21)
\put(0,8){\line(1,0){12}}
\put(2,0){\line(1,0){8}}
\put(2,0){\line(-1,4){2}}
\put(10,0){\line(1,4){2}}
\qbezier(6,8)(4,13)(8,21)
\qbezier(5.3,11)(3,14)(4,17)
\qbezier(5.5,14)(6,16)(10,18)
\end{picture}}
\put(25,0){\begin{picture}(17,14.5)
\put(7,0){\framebox(9,12){}}
\put(7,8){\line(-6,5){6}}
\put(7,6){\line(-1,1){7}}
\put(0,13){\line(1,0){1}}
\put(16,7){\oval(4,6)[r]}
\put(11.5,12){\oval(6,3)[t]}
\put(11.5,13.5){\oval(1.5,2)[t]}
\end{picture}}
\end{picture}
\end{center}
```

输出为



子图可以保存在一个盒子中, 以后可以整体调用这个盒子, 不必再一一输入各条命令. 这个盒子称为子图盒子. 要保存和重复使用子图, 首先应创建子图盒子名称:

```
\newsavebox{\子图盒子}
```

然后用下述命令将部分图形保存在名为子图盒子的子图盒子中:

```
\savebox{\子图盒子}(宽度,高度)[位置]{子图图形}
```

其中参数(宽度,高度)[位置]的含义与第90页\makebox中的含义一样. 参数子图图形可以是文本, 也可以是一个子图环境, 或者是一些绘图命令.

以后可以把命令

```
\usebox{\子图盒子}
```

当作图形元素放置在其它图形中的任何地方.

如果\savebox定义在一个环境中, 那么随着环境的结束它的值也就消失了. 如果定义在导言区, 那么整篇文档都可使用这个子图盒子. 但要注意, 子图盒子要占用分配给TeX的内存空间, 所以当不再使用子图盒子时, 可将其内容重设为空, 简单的办法是使用命令:

```
\sbox{\子图盒子}{}%
```

下面是使用子图盒子的例子, 首先定义子图盒子名称, 然后将子图保存在这个盒子中, 这是由下述语句完成的:

```
\newsavebox{\flower}
\savebox{\flower}(12,21){%
\begin{picture}(12,21) \thicklines
\put(0,8){\line(1,0){12}} \put(2,0){\line(1,0){8}}
\put(2,0){\line(-1,4){2}} \put(10,0){\line(1,4){2}}
\qbezier(6,8)(4,13)(8,21) \qbezier(5.3,11)(3,14)(4,17)
\qbezier(5.5,14)(6,16)(10,18) \end{picture}}
```

下面是在一个图形环境中使用子图盒子(见7-3-2.tex).

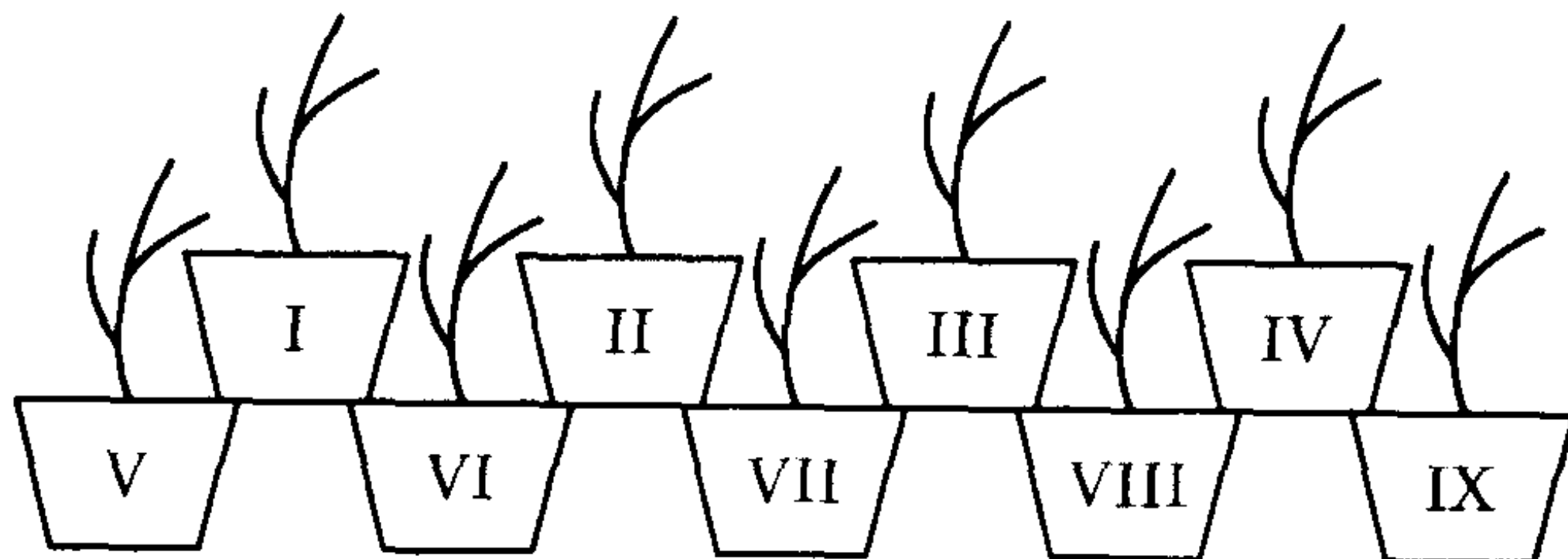
```
\newcommand{\wrt}[1]{\makebox(0,0)[c]{#1}}
\newcounter{pot}\setcounter{pot}{0}
```

```

\begin{center}
\begin{picture}(84,29)
\multiput(0,0)(18,0){5}{\usebox{\flower}}
\multiput(9,8)(18,0){4}{\usebox{\flower}}
\multiput(15,12)(18,0){4}{\addtocounter{pot}{1}\wrt{\Roman{pot}}}
\multiput(6,4)(18,0){5}{\addtocounter{pot}{1}\wrt{\Roman{pot}}}
\end{picture}
\end{center}
\sbox{\flower}{}

```

为了在图形中显示顺序数字, 设置了一个计数器`pot`, 每用一次子图就使计数器加1, 并且用大写罗马数字显示计数器的值. 显示结果为



因为以后不再使用子图盒子`\flower`, 所以最后用一条命令释放子图盒子占用的内存.

子图不一定是用子图环境生成的, 也可以是单独的绘图命令. 下面例子定义了几个子图盒子, 存放的子图分别是流程图中的处理框、条件判断框和输入输出框, 这3个子图的坐标原点都放在了子图的中心. 另外定义了几个命令简化水平和竖直方向的直线和矢量线的输入. 源文件是(见7-3-3.tex):

```

\newcommand{\wrt}[1]{\makebox(0,0)[c]{#1}}
\newcommand{\lline}[1]{\line(-1,0){#1}}
\newcommand{\rline}[1]{\line(1,0){#1}}
\newcommand{\uline}[1]{\line(0,1){#1}}
\newcommand{\dline}[1]{\line(0,-1){#1}}
\newcommand{\lvec}[1]{\vector(-1,0){#1}}
\newcommand{\rvec}[1]{\vector(1,0){#1}}
\newcommand{\uvec}[1]{\vector(0,1){#1}}
\newcommand{\dvec}[1]{\vector(0,-1){#1}}
\newsavebox{\condition}

```

```

\newsavebox{\process}
\newsavebox{\inputoutput}
\savebox{\process}(0,0){\thicklines
  \put(-18,-3){\framebox(36,6){}}
}
\savebox{\condition}(0,0){\thicklines
  \put(-10,0){\line(2,1){10}}
  \put(-10,0){\line(2,-1){10}}
  \put(10,0){\line(-2,1){10}}
  \put(10,0){\line(-2,-1){10}}
  \put(-10,0){\lline{10}}
  \put(-15,1){\makebox(0,0)[b]{no}}
  \put(10,0){\rline{10}}
  \put(15,1){\makebox(0,0)[b]{yes}}
}
\savebox{\inputoutput}(0,0){\thicklines
  \put(-19.5,-3){\rline{36}}
  \put(-19.5,-3){\line(1,2){3}}
  \put(19.5,3){\lline{36}}
  \put(19.5,3){\line(-1,-2){3}}
}
\begin{center}
\begin{picture}(80,63)(0,-63)\thicklines
\put(40,0){\dvec{5}}
\put(40,-10){\usebox{\condition}}
\put(40,-10){\wrt{\Delta\ne 0?}}
\put(20,-10){\dvec{7}}
\put(20,-20){\usebox{\process}}
\put(20,-20){\wrt{\mathbf{x}_{1,2}=-b/2}}
\put(60,-10){\dvec{7}}
\put(60,-20){\usebox{\process}}
\put(60,-20){\wrt{\mathbf{p}=-b/2,\mathbf{q}=\sqrt{\Delta}}}
\put(60,-23){\dvec{7}}
\put(60,-33){\usebox{\process}}

```

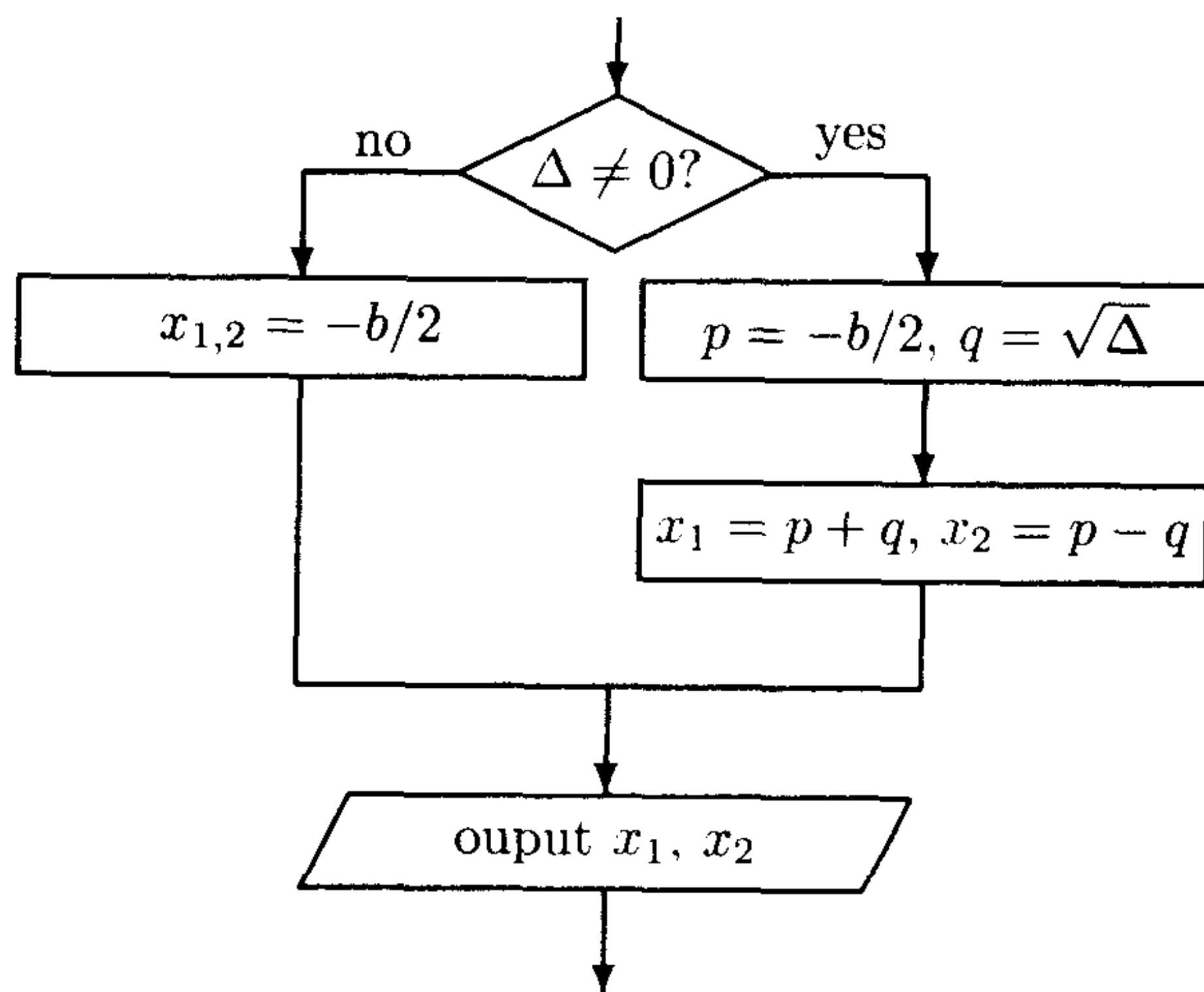


```

\put(60,-33){\wrt{$x_1=p+q,\,x_2=p-q$}}
\put(20,-23){\dline{20}}
\put(60,-36){\dline{7}}
\put(20,-43){\rline{40}}
\put(40,-43){\dvec{7}}
\put(40,-53){\usebox{\inputoutput}}
\put(40,-53){\wrt{ouput $x_1,\,x_2$}}
\put(40,-56){\dvec{7}}
\end{picture}
\end{center}

```

输出为



§7.4 天元的绘图功能

Tydraw 是肖刚设计的在 T_EX 环境里使用的绘图软件, 其特点是能利用 T_EX 画出精确的插图. 经陈志杰修改, 又增加了一些新的功能.

如果在源程序中使用了 Tydraw, 则必须在前面 (例如在导言区) 加入语句 `\input tdw`. Tydraw 最好嵌入在 L^AT_EX 的 picture 环境中使用, 因为 Tydraw 使用的长度单位是 1mm, 当 `\unitlength` 被设定为 1mm 后, Tydraw 与 picture 环境配合得天衣无缝. 因此最好在文件的导言区加入命令:

```
\setlength{\unitlength}{1mm}
```

在 picture 环境中, Tydraw 命令块的整体结构如下:

$$\backslash\text{put}(a,b)\{\backslash\text{draw } c,d,\{\dots\}\}$$

这里 a, b, c, d 应该是 4 个数, Tydraw 所绘图形的参考点被安放在 picture 环境的坐标为 (a, b) 的点上, 而 $\{\dots\}$ 中的各条指令的原点则定位在相对于 Tydraw 所绘图形的参考点(左下角)的坐标为 (c, d) 的点上. 因此 Tydraw 图形原点在 picture 环境里的坐标是 $(a+c, b+d)$, 特别当 a, b, c, d 都取 0 时, Tydraw 的坐标系与 picture 环境里的坐标系一致.

$\{\dots\}$ 中的 Tydraw 命令如以下所列(注意所有的命令均不含 '\', 相互之间用逗号 ',' 分隔). 命令中的参数都可以使用表达式. 所谓表达式是指可使用符号: $+$, $-$, $*$, $/$ 表示加减乘除(当无歧义时, 乘号 $*$ 可省略, 例如 $100\sin(t)$); $^$ 表乘幂(注意底数不能等于 0, 当指数不是整数时, 底数必须是正数). 指数运算的级别最高, 然后是乘法和除法, 最后才是加减法, 与通常代数运算的惯例相同. $|\dots|$ 表绝对值, e 和 π 代表常数 e 和圆周率. 用 \sin , \cos , tg , \arcsin , \arccos , \arctg , \exp , \ln , \lg , sqrt , sh , ch , th 表示各种初等函数. 如果对运算的先后次序没有把握时, 可使用圆括号“(”与“)”, 但不能使用方括号或花括号.

Tydraw 命令:

1. 设定线条宽度: $\text{width}(w)$ $w=0, 1, 0.2, \dots, 1.0$, 一般可使用 0.2. 此命令可多次使用. 默认值为 0.1.
2. 画直线: $\text{line}(x_1, y_1, x_2, y_2)$ 从 (x_1, y_1) 到 (x_2, y_2) 的直线.
3. 画平移直线: $\text{lines}(x_1, y_1, x_2, y_2, dx, dy, n)$ 从直线 $(x_1, y_1)-(x_2, y_2)$ 开始, 以 (dx, dy) 做平移向量, 画 n 条直线. 注意这里的 n 应是一个正整数.
4. 画折线: $\text{polygon}(x_1, y_1, x_2, y_2, \dots, x_n, y_n)$.
5. 画箭头: $\text{arrowhead}(x, y, L, d_1, d_2)$ 或 $\text{arrowheadd}(x, y, dx, dy, L, d_2)$ 以 (x, y) 作为箭尖, 画一个指向 d_1 度(或方向向量为 (dx, dy))的箭头, 箭头的边长为 L , 箭头的边与中心线的夹角为 d_2 度. 推荐值: $L=2, d_2=10$.
6. 画圆: $\text{circle}(x, y, r)$ 圆心在 (x, y) , 半径为 r .
7. 点: $\text{point}(x, y, d)$ d 为直径, $d \leq 18.8$.
8. 画长方形: $\text{box}(x_1, y_1, x_2, y_2)$ 以 $(x_1, y_1), (x_2, y_2)$ 作为两个顶点的空心矩形.
9. 实心矩形: $\text{fill}(x_1, y_1, x_2, y_2)$.
10. 参数方程定义的曲线: $\text{trace}(t_1, t_2, f_1(t), f_2(t))$ 这条命令是 Tydraw 的核心, 因为极大部分绘图软件都无法画出这类曲线, 这里曲线的参数方程为: $x=f_1(t), y=f_2(t), t_1, t_2$ 界定参数 t 的取值范围.
11. 虚线: $\text{dotted}(n, p)$ 此命令以后的图形都被画成虚线. 如要再恢复实线模式, 可使用命令 $\text{dotted}(0, 0)$. 其中 $n \leq 16$, 表示虚线的周期, 其长度单位是 1pt 左

右. p 的二进制表示式就是每个周期内虚线的模式, 数字 1 表示黑点, 数字 0 表示空白. 当 $(n,p)=(6,7)$ 时, 因为 7 的二进制表达式是 111, 因此这条虚线是 3pt 长的短线, 互相间隔 $6-3=3\text{pt}$. 下面列出一些不同参数虚线的样本.

<code>dotted(2,1)</code>	-----
<code>dotted(4,3)</code>	-----
<code>dotted(5,7)</code>	-----
<code>dotted(6,7)</code>	-----
<code>dotted(15,16381)</code>	-----

12. spline 曲线: `acurve(x1,y1,d1,...)` 以及 `curve(x1,y1,d1,...)` 规定在 (x_i,y_i) 处的切线角度为 d_i 度. 这两条命令画出的曲线略有不同, 用户可比较采用合适的一种.
13. 消隐: `hide(b1,L1,b2,L2,...)` 这条命令仅对紧跟在后面的命令有效. 表示在参数取值为 b_1, b_2, \dots 的地方消隐长度为 L_1, L_2, \dots 的弧. 最多消隐 32 处. 这里直线或 spline 线段的参数取值从 0 至 1, 圆周则从 0 至 2π .
14. 格点: `lattice(x0,y0,dx1,dy1,dx2,dy2,n1,n2,d)` 画 $n_1 \times n_2$ 个点, 以 (x_0,y_0) 作为起点, 以 (dx_1,dy_1) 和 (dx_2,dy_2) 作为生成向量, d 是点的直径.
15. 网格: `net(t1,t2,u1,u2,f1(t,u),f2(t,u))` 画出具有两个参数的曲面网格: $x=f_1(t,u), y=f_2(t,u)$. 网格的步长 (即参数 t, u 的增量) 都是 1.
16. 画阴影线: `shadow(t1,t2,dt,f1(t),g1(t),f2(t),g2(t))` 画出从点 $(f_1(t), g_1(t))$ 到点 $(f_2(t), g_2(t))$ 的直线, 其中参数 t 的取值从 t_1 至 t_2 , 以 dt 作为增量.
17. 缩放比: `ratio(r)` 默认的比值 $r=1$. 此命令设定 r 的值. 在此以后的命令中的参数均被放大 r 倍. 不过所有的角度都不受影响, `width`, `dotted`, `hide` 命令也不受影响. 此外 `lines` 中的 n , `arrowhead` 与 `arrowheadd` 中的 L , `point` 中的 d , `lattice` 中的 n_1, n_2, d , 以及 `trace`, `net`, `shadow` 中的参数变化范围 t_1, t_2 均不受影响, 但参数方程的值都被放大 r 倍, 因此整个图形被放大了 r 倍, 如果你同时重新规定 `\unitlength` 的值为 $r \text{ mm}$, 就能做到与 LaTeX 的图形同时放大.
18. 标注文字: `write(x,y)[位置]{标注内容}` 这条命令利用 `\makebox(0,0)[...]{...}` 把文字标注在图形上, 因此位置可选取 c, l, r, t, b 及其组合. 当 `ratio` 改变时, 这些文字的位置也会同步改变.
19. 三维曲线: `threed(θ, ϕ)` (θ, ϕ) 就是 Maple 立体图的视角: θ 是视线在 xOy 平面上的射影与 x 轴的夹角, ϕ 是视线与 z 轴的夹角. 例如 Maple 默认的视角

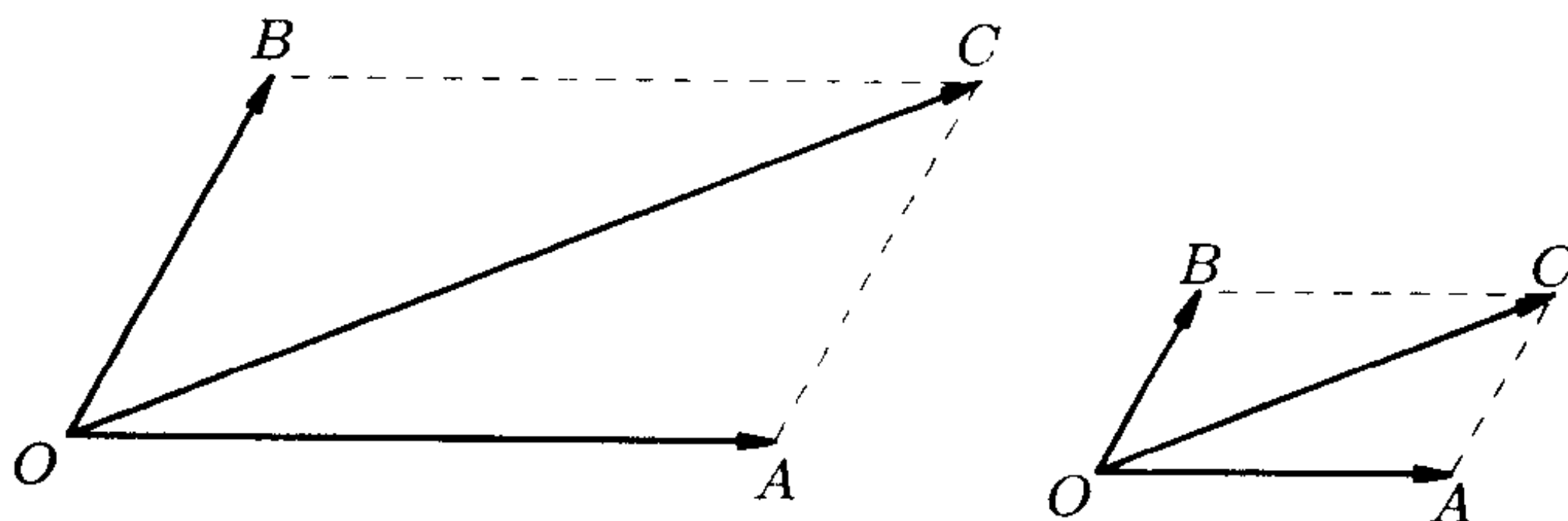
是 (45, 45), 正等测投影的视角是 (45, 54.7), 正二等测投影的视角是 (20.7, 70.5) 等. 从这条命令开始, 下列命令:

```
line(x1,y1,z1,x2,y2,z2)
lines(x1,y1,z1,x2,y2,z2,dx,dy,dz,n)
polygon(x1,y1,z1,x2,y2,z2,...,xn,yn,zn)
arrowhead(x,y,z,L,d1,d2)
arrowheadd(x,y,z,dx,dy,dz,L,d2)
circle(x,y,z,d)
point(x,y,z,r)
trace(t1,t2,f1(t),f2(t),f3(t))
write(x,y,z)[.]{...}
```

都变成了三维的平面图. 其他命令则不受影响. 如果要恢复平面图形的模式, 可使用命令 `twod`. 读者将会发现要改变视角十分容易, 唯一受影响的是曲线被遮盖的部分, 需要重新调整, 此外, 立体图形的包络也可能变化.

与 $\text{T}_{\text{E}}\text{X}$ 一样, `%` 被用来作为注解号, 从这个字符开始直至行尾都被忽略. 不过注解符 `%` 必须出现在完整的 `Tydraw` 命令的前面或后面, 如果出现在一条命令的中间, 就会出错.

我们先举一个向量相加的平行四边形法则的例子 (参见 7-4-1.ty).



上图的源程序如下:

```
\begin{center}
\setlength{\unitlength}{1mm}
\begin{picture}(56,26)(-3,-3)\small
\put(0,0){\draw0,0,{width(.3),polygon(39,0,0,0,11,20),
line(0,0,50,20),arrowheadd(39,0,1,0,2,10),
arrowheadd(50,20,50,20,2,10),
arrowheadd(11,20,11,20,2,10),
dotted(6,7),width(0.1),polygon(39,0,50,20,11,20),
```



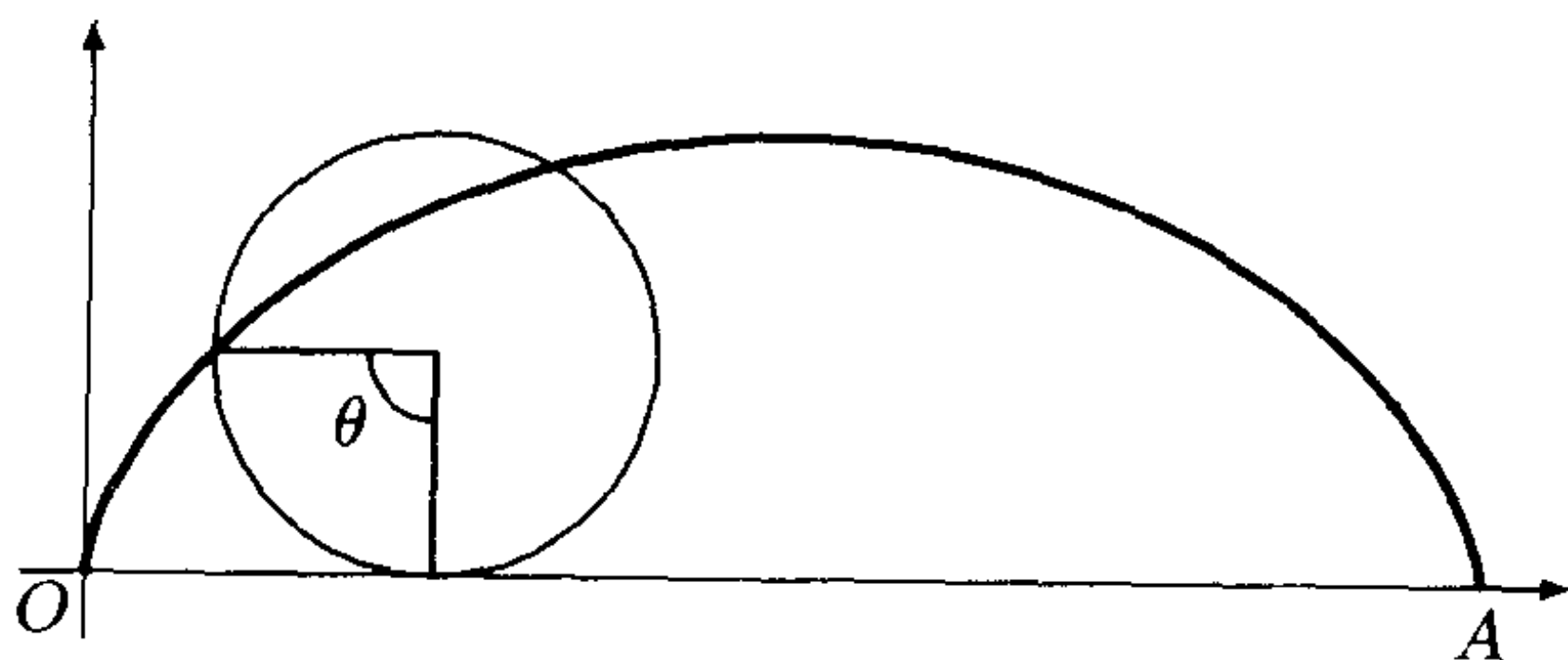
```

write(-0.5,-0.5)[rt]{$O$},write(39,-1)[t]{$A$},
write(11,21)[b]{$B$},write(50,21)[b]{$C$}}
\end{picture}
\setlength{\unitlength}{0.5mm}
\begin{picture}(56,26)(-3,-3)\small
\put(0,0){\draw0,0,{width(.3),ratio(0.5),
.....
\end{picture}
\end{center}

```

由于使用了 Tydraw 命令 `ratio(0.5)` 以及把 `\unitlength` 设定为 0.5 mm, 不需改动源程序的其余部分, 就能使右边的图成为左边的一半大小. 而命令 `\small` 的作用是使得标注的字母变得小一些.

陈志杰主编的《高等代数与解析几何》中的插图都是用 Tydraw 画的. 下面是摆线图形及其 Tydraw 源程序(参见 7-4-2.ty).



```

\begin{picture}(70,28)(-3,-3)\small
\put(-3,0){\vector(1,0){70}}
\put(0,-3){\vector(0,1){28}}
\put(0,0){\draw0,0,{width(0.4),
trace(0.01,2pi-0.01,10(t-sin(t)),10(1-cos(t))),
width(0.2),
circle(15.7,10,10),polygon(5.7,10,15.7,10,15.7,0),
trace(pi,1.5pi,15.7+3cos(t),10+3sin(t)),
write(-0.5,-0.5)[rt]{$O$},write(20pi,-1)[t]{$A$},
write(12.7,8)[rt]{$\theta$}}}
\end{picture}

```

摆线的一个拱的参数范围应该是 0 到 2π , 但是 0 与 2π 是摆线的奇点, Tydraw 有可能出错, 因此 `trace` 命令的参数范围改成 0.01 到 $2\pi - 0.01$, 以避免奇点.

下面是绘制球面的立体图的例子(参见7-4-3.ty), 左右两个图采用了不同的视角, 左边是正等测投影的视角($45^\circ, 54.7^\circ$), 右边是 Maple 默认的视角($45^\circ, 45^\circ$), 除了 \threed 的参数不同外, 其余命令完全相同, 可见改变立体图的视角十分容易. 当然读者会发现右边图形的虚实线画得不对, 这可通过试验来调整.

```

\begin{center}
\begin{picture}(44,44)(-20,-20)
\put(0,0){\draw0,0,{width(.4),
circle(0,0,20),
threed(20.7,70.3),
trace(-69.3*pi/180,110.7*pi/180,20cos(t),20sin(t),0),
trace(-0.366,pi-0.366,20sin(t),0,20cos(t)),
trace(-0.792,pi-0.792,0,20sin(t),20cos(t)),
line(20,0,0,28,0,0),
line(0,20,0,0,24,0),
line(0,0,20,0,0,24),
arrowhead(28,0,0,1,0,0,2,10),
arrowhead(0,24,0,0,1,0,2,10),
arrowhead(0,0,24,0,0,1,2,10),
write(29,0,0)[rt]{$x$},
write(0,25,0)[l]{$y$},
write(0,0,25)[b]{$z$},
write(0,0,-0.5)[lt]{$0$},
dotted(6,7),width(.2),
trace(110.7*pi/180,290.7*pi/180,20cos(t),20sin(t),0),
trace(2pi-0.366,pi-0.366,20sin(t),0,20cos(t)),
trace(2pi-0.792,pi-0.792,0,20sin(t),20cos(t)),
line(-20,0,0,20,0,0),
line(0,-20,0,0,20,0),
line(0,0,-20,0,0,20)}}
\end{picture}
\quad
\begin{picture}(44,44)(-20,-20)
\put(0,0){\draw0,0,{width(.4),
circle(0,0,20),

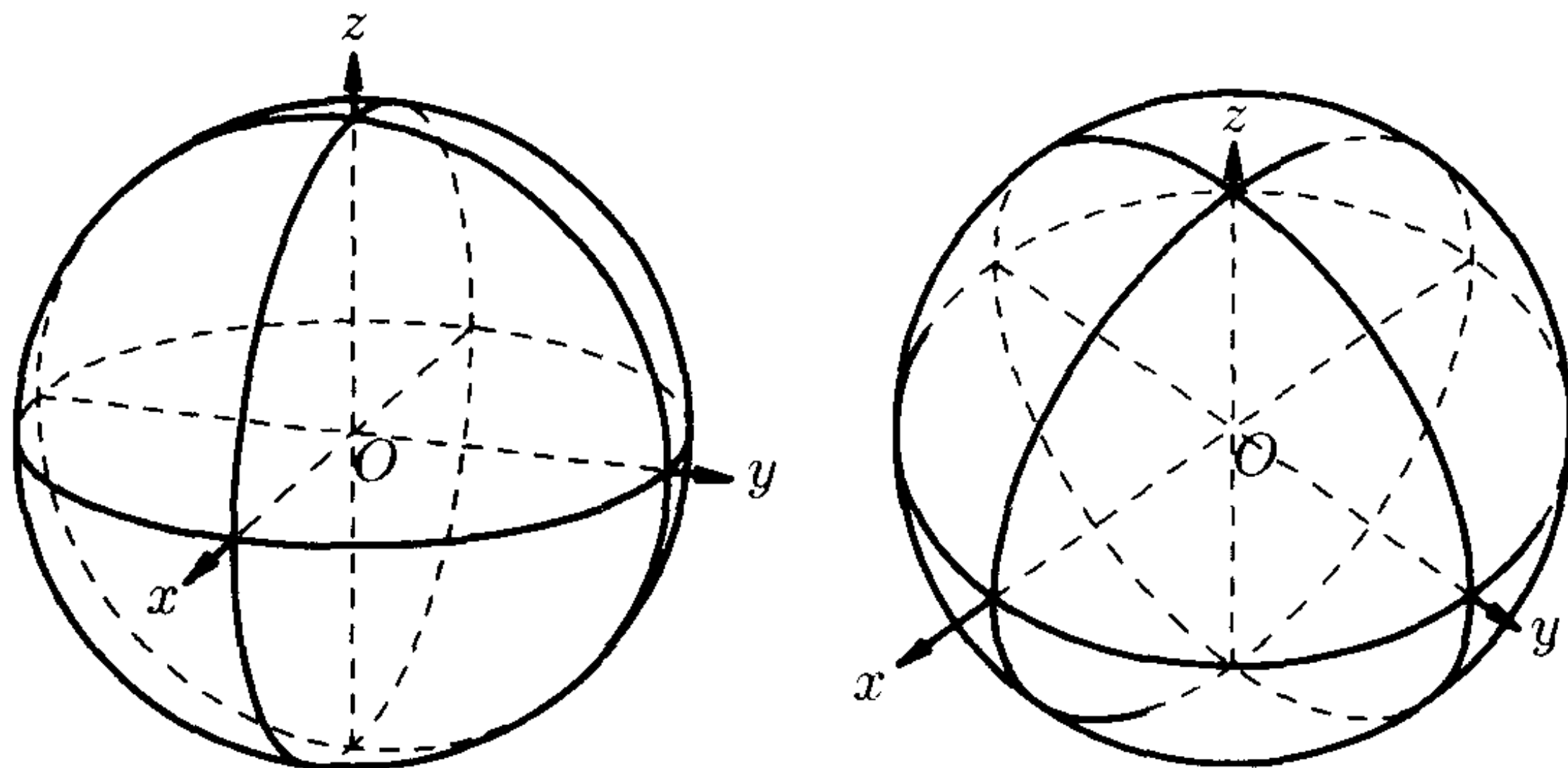
```

```

threed(45,45),
.....
\end{picture}
\end{center}

```

输出为



§7.5 插入外部图形

外部图形的格式多种多样, 显示或打印时不同格式需要不同的驱动程序, 故而依赖于外部设备, 因此在 $\text{T}_{\text{E}}\text{X}$ 中不能直接处理外部图形. 当初 Knuth 设计 $\text{T}_{\text{E}}\text{X}$ 时提供了一条 `\special` 命令, 通过它可以向 `dvi` 文件传送各种信息, 例如关于外部图形位置和名字的信息, 这些信息是 $\text{T}_{\text{E}}\text{X}$ 本身所不理解的, 要由具体的 `dvi` 预览或打印设备来解释和处理这些信息. 本来 `dvi` 文件是和设备无关的, 但它含有 `\special` 命令后, 实际上就和设备驱动程序有关了.

本节中提到的在 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 文件中“插入图形”均指在文件中直接或间接使用了 `\special` 命令, 使得在最后生成的 `dvi` 文件中插入了外部图形.

7.5.1 使用 $\text{emT}_{\text{E}}\text{X}$ 时插入图形

在 DOS 环境中最有名的 $\text{T}_{\text{E}}\text{X}$ 系统是 $\text{emT}_{\text{E}}\text{X}$. 在这个系统中, 使用 `\special` 命令, 可以向 $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$ 文件插入外部图形, 但仅限于黑白的 `bmp` 和 `pcx` 图形. 例如把一个宽 5cm 高 3cm 的图形 `D:\pic\tu1.bmp` 插入在当前位置, 可使用如下语句:

```

\setlength{\unitlength}{1mm}
\begin{center}\begin{picture}(50,30)
\put(0,30){\special{em:graph D:/pic/tu1.bmp}}
\end{picture}\end{center}

```

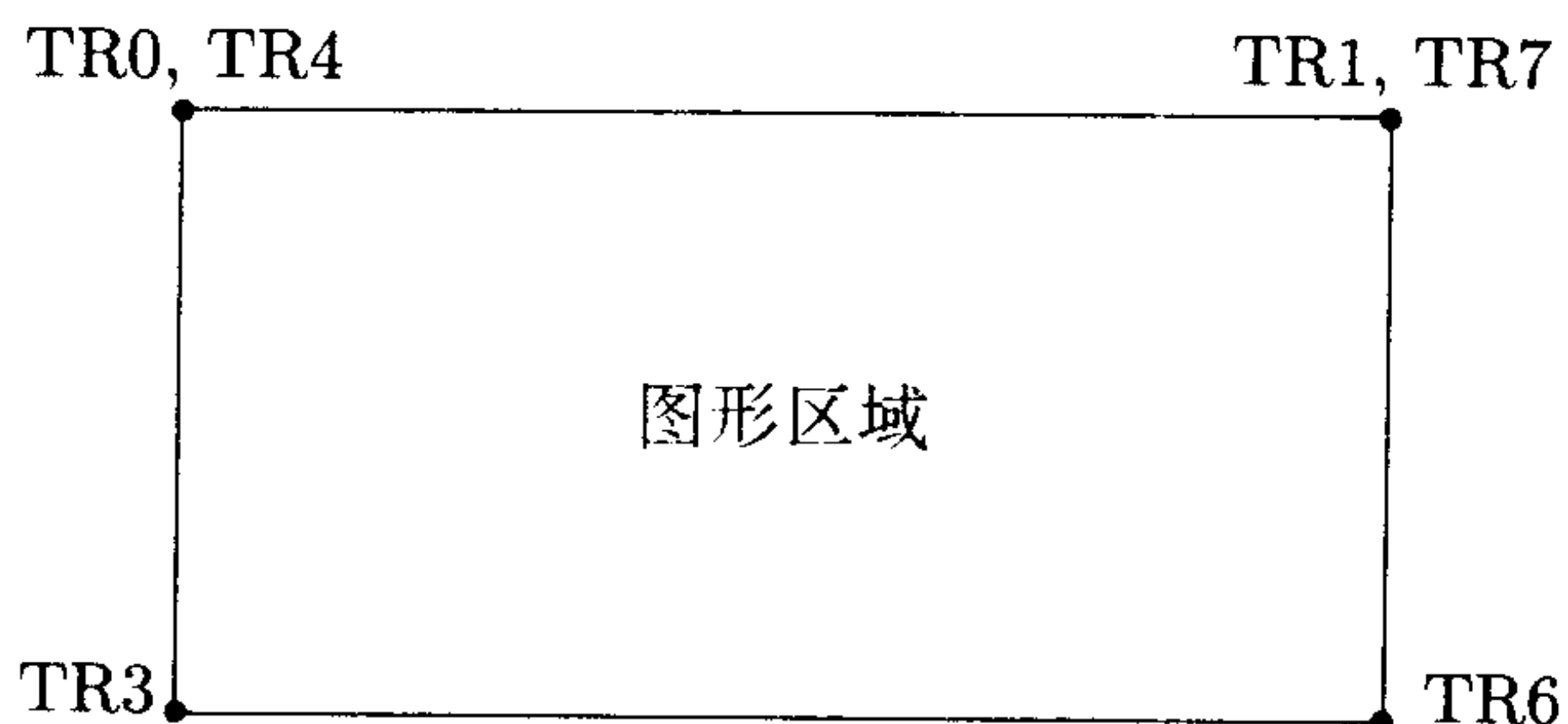
从上例可以看出, 在 emTeX 环境中, 插入外部图形时, 应将插入基准点选在图形环境确定的区域的左上角而非左下角, 此外文件路径中的分隔符不是倒斜线, 而是 Unix 风格的“从东北到西南”的斜线.

还要说明的一点是: 对于像 `bmp` 和 `pcx` 这类点阵图形, emTeX 的预览与打印驱动程序都没有缩放功能, 因此最终得到的图形大小与预览或打印使用的分辨率有关. 例如用分辨率为 600DPI (DPI = dots per inch, 每英寸的打印点数) 的打印机打印出一个 $5\text{cm} \times 3\text{cm}$ 的图形 `tu1.bmp`, 那么 `tu1.bmp` 在长宽两个方向上的像素个数分别为

$$\frac{5}{2.54} \times 600 \approx 1181, \quad \frac{3}{2.54} \times 600 \approx 709.$$

如果用分辨率为 300DPI 的打印机打印同一个图像文件 `tu1.bmp`, 最终得到的图像尺寸将是 $10\text{cm} \times 6\text{cm}$!

有时为了印刷制版的需要, 要将 `dvi` 文件“反打” (打印镜像图形), emTeX 环境下一些 `dvi` 文件预览器的 `transformation` 命令可以实现反打功能, 但对插入到 `dvi` 文件中的图形不作任何变换. 为反打外部图形, 需事先用图形处理软件反转图形 (应对应于 `transformation` 的反转方式), 存为新的文件, 在 $\text{L}^{\text{A}}\text{TeX}$ 源文件中使用这个新文件并改用合适的插入基准点. 下图示例了对于 `transformation` 的不同值, 插入点在图形区域上的位置.



从上可知, 使用 emTeX 出书时, 应为每个插图准备 3 个文件: 一个用于屏幕显示 (分辨率较低), 一个用于打印校样 (分辨率较高), 一个镜像图形用于制版. 笔者之一编写的《常用软件操作指南》一书使用天元和 emTeX 排版, 其中的插图都是如上处理的.

7.5.2 使用 PCTeX32 时插入图形

在 Windows 环境中, PCTeX32 支持 4 种图形文件格式: Windows bitmap (BMP), Encapsulated PostScript (EPS), Windows Metafile (WMF 或 EMF) 以及 PostScript (PS). 用法是

```

\special{bmp:文件标识符 x=宽度 y=高度}
\special{eps:文件标识符 x=宽度 y=高度}
\special{wmf:文件标识符 x=宽度 y=高度}
\special{ps:文件标识符 x=宽度 y=高度}

```

其中文件标识符由盘符、路径和文件名组成. 宽度和高度不是图形文件原有的尺寸, 而是想要它变成的尺寸(这会产生放缩效果), 可用 cm 或 in 作单位. 例如想把图形 D:\pic\tu1.bmp 居中插入在当前位置, 并把它变为宽 3cm 高 2cm 的图形, 可使用语句:

```
\vspace*{2cm}
```

```
\centerline{\special{bmp:D:/pic/tu1.bmp x=3cm y=2cm}}
```

或使用绘图环境:

```
\setlength{\unitlength}{1mm}
```

```
\begin{center}\begin{picture}(30,20)
```

```
\put(0,0){\special{bmp:D:/pic/tu1.bmp x=3cm y=2cm}}
```

```
\end{picture}\end{center}
```

使用 `\vspace*{...}` 或使用绘图环境都是为了留出图形所占位置. 使用绘图环境的好处是既可以插入外部图形又可以绘制内部图形. 注意在绘图环境中外部图形基准点是左下角, 这与 emTeX 中的用法是不同的.

PCTeX32 提供了宏包 `setbmp`, `seteps`, `setwmf` 和 `setps`, 使用 `\input` 引入相应宏包后, 可用下述命令简化输入:

```

\setbmp{缩进量}{宽度}{高度}{文件标识符}
\seteps{缩进量}{宽度}{高度}{文件标识符}
\setwmf{缩进量}{宽度}{高度}{文件标识符}
\setps{缩进量}{宽度}{高度}{文件标识符}

```

其中缩进量是插入的图形与页面左边界的距离. 若居中放置图形, 可用下述命令:

```

\centerbmp{宽度}{高度}{文件标识符}
\centereps{宽度}{高度}{文件标识符}
\centerwmf{宽度}{高度}{文件标识符}
\centerps{宽度}{高度}{文件标识符}

```

除非想在图形的上方或下方留出额外的空白, 上述 8 条命令的前后不必再连用 `\vspace*{...}` 命令.

7.5.3 图形包

从上一节可知, 在dvi文件中插入外部图形可以用原始的`\special`命令, 但在不同的系统中, 或者说使用不同的驱动程序时, 具体的用法是不同的. 源文件不能通用, 这是很不方便的.

在 \LaTeX 2_ϵ 中已开发出了几个图形宏包, 如“标准”的`graphics`宏包和“扩展”的`graphicx`宏包. 常用的还有`color`宏包. 利用这些宏包可以容易地插入外部彩色或黑白图形, 并可进行裁剪、放缩、翻转和旋转. 在`graphics`和`graphicx`中, 定义的命令和命令的不可省略参数都是相同的, 差别仅在于可选项的格式和内容. 两个宏包为所有的驱动程序提供了一组通用的命令, 而与驱动程序有关的代码存放在特定的`def`文件中, 通过宏包的选项加载它们, 因此只要改变选项, 就可以切换到另一种驱动程序, 正文本身无需修改. 因为标准的`graphics`宏包更符合 \LaTeX 的语法, 所以介绍这个宏包, 关于宏包`graphicx`的介绍可参见第222页. 使用该宏包时, 应先在导言区写上

```
\usepackage[选项]{graphics}
\usepackage[选项]{color}
```

其中选项处应包含驱动程序名, 可用的有:

<code>dvipdf</code>	<code>dviwindo</code>	<code>pctexhp</code>	<code>tcidvi</code>
<code>dvips</code>	<code>emtex</code>	<code>pctexps</code>	<code>textures</code>
<code>dvipsone</code>	<code>oztex</code>	<code>pctexwin</code>	<code>truettex</code>
<code>dviwin</code>	<code>pctex32</code>	<code>pdftex</code>	<code>xdvi</code>

其中`dviwin`, `emtex`, `pctexhp`, `pctexwin`不支持彩色、放缩和旋转; `truettex`不支持放缩和旋转. 多数情况下可使用`dvips`选项.

含有中文的天元源文件编译成dvi文件后, 显示或打印时仍需要中文环境. 在生成dvi文件后将其转换成PostScript文件, 就可脱离中文环境, 在任何支持ps文件的设备上显示或打印排版结果. 如果使用选项`dvips`, 则将dvi文件转换成ps文件后, 可使得插入的外部图形也可在支持ps文件的设备上显示或打印出来. WinEdt或WinShell集成环境含有工具按钮可很方便地将dvi文件转换成ps文件. 除非特别说明, 本节总是假设在调用图形宏包时使用了`dvips`选项.

除了驱动程序名, 用于`graphics`宏包的其他的一些选项有:

`draft` 不插入图形, 仅保留出图形所占位置, 显示图形边界方框及图形文件名. 这可明显加快处理速度.

final 真正插入图形. 这是默认值. 但若在\documentclass 选项中使用了 draft, 则必须显式写上选项 final 才能真正插入图形.

hidescale 当对图形放缩时, 只保留放缩后所占区域而不插入图形. 当预览器不支持放缩时必须使用该选项.

hiderotate 当对图形旋转时, 只保留旋转后所占区域而不插入图形. 当预览器不支持旋转时必须使用该选项.

hiresbb 在插入的 eps 图形文件中不使用通常的 %%BoundingBox 的值, 而是用 %%HiresBoundingBox 的值作为图边界盒子的值. 边界盒子(BoundingBox)是包含图形的正矩形区域, 正矩形是四边横平竖直的矩形, 不是倾斜的矩形.

monochrome 用于 color 宏包的选项, 使彩色命令不起作用. 当预览器不支持彩色时应使用该选项.

当两个宏包的选项完全相同时, 例如都只有一个驱动程序选项, 可将它们放在同一个\usepackage 命令里.

图形宏包支持最好的图形格式是 EPS (Encapsulated PostScript) 格式, 本节就以这种格式的图形文件作例子. 将其它格式的图形文件转换成 EPS 格式是很容易的, 很多软件都可进行图形格式转换, 例如功能很强的免费软件 ImageMagick, 可从 www.imagemagick.org 或 ftp.wizards.dupont.com 等站点自由下载.

EPS 图形文件的头部形如

```

%!PS-Adobe-3.0 EPSF-3.0
%%Creator: Adobe Photoshop Version 6.0
%%Title: panda2.eps
%%CreationDate: Sat Oct 27 2001 17:16:27
%%BoundingBox: 0 0 59 59
%%HiResBoundingBox: 0 0 59.25 59.25
%%SuppressDotGainCompensation
%%DocumentProcessColors: Black
%%EndComments

```

其中边界盒子(BoundingBox)的数据反映了图形的大小. 上例中的数据是 0 0 59 59, 表示边界盒子左下角坐标为 (0,0), 右上角坐标为 (59,59), 单位是“大点” bp (big point), 1in=72bp=72.27pt. 容易算出这个图形的宽与高均为 59bp, 约为 0.819 英寸, 即 2.08 厘米.

图形盒子左下角坐标不一定都是 (0,0), 例如文件 pic.eps 的头部几行如下:

```

%!PS-Adobe-2.0 EPSF-1.2
%%Creator: MATLAB, The MathWorks, Inc
%%Title: MATLAB graph
%%CreationDate: 03/09/98 17:26:21
%%Pages: 001
%%BoundingBox: 080 407 532 756
%%DocumentFonts: Times-Roman
%%DocumentNeededFonts: Times-Roman
%%EndComments

```

边界盒子左下角坐标为 (80, 407), 右上角坐标是 (532, 407), 图形宽度应是 $532 - 080 = 452$ bp, 高度是 $756 - 407 = 349$ bp. 可以想象有一个坐标系, 边界盒子的坐标以及下文提到的剪裁坐标都是相对于这个坐标系的. 这个坐标系的原点实际是纸张的左下角.

7.5.4 插入图形的基本命令

插入图形的基本命令是

```

\includegraphics*[左下角x,y][右上角x,y]{图形文件}
\includegraphics[左下角x,y][右上角x,y]{图形文件}

```

其中带 * 号的命令实际插入剪裁后的图形, 剪裁区域的左下角与右上角的坐标由命令中的可选项决定. 如果省略了左下角坐标, 则默认左下角坐标取 (0, 0). 如果不写可选项, 则通过读取边界盒子的值确定图形大小. 坐标可以带单位, 无单位时默认单位是 bp. 上述坐标仅仅是用于确定剪裁区域的大小及在原来图形上的位置, 当将其插入到文档时, 是插在当前位置, 与原来的坐标系没有关系.

不带 * 号的命令是插入整个图形, 剪裁范围的坐标仅仅用于确定图形占用的区域, 当剪裁范围与图形边界盒子不相符时, 有可能使插入的图形与周边文字重叠或出现过多的空白.

有些驱动程序不支持图形的剪裁, 或者不支持某些格式图形的剪裁, 此时上述两个命令有可能产生意外的效果, 比如剪裁不起作用, 总是插入整个图形, 或者将整个图形放缩到剪裁区域的大小. 后面介绍的其他命令也会有类似问题, 不再一一赘述.

下面是一个例子 (参见 7-5-1.tex), 特意加了方框以看出剪裁区域的范围:

```

{\setlength{\fboxsep}{0pt}

```

```

\begin{center}
\fbbox{\includegraphics*[10,10][30,40]{pic/panda.eps}}\quad
\fbbox{\includegraphics[10,10][30,40]{pic/panda.eps}}\quad
\fbbox{\includegraphics{pic/panda.eps}}
\end{center}

```

排版输出结果是



可以看出, 对齐位置和相互间距都是取决于裁剪区域的.

7.5.5 放大和缩小

按比例进行放缩的命令是

```

\scalebox{横向放缩因子}[竖向放缩因子]{插入的图形}

```

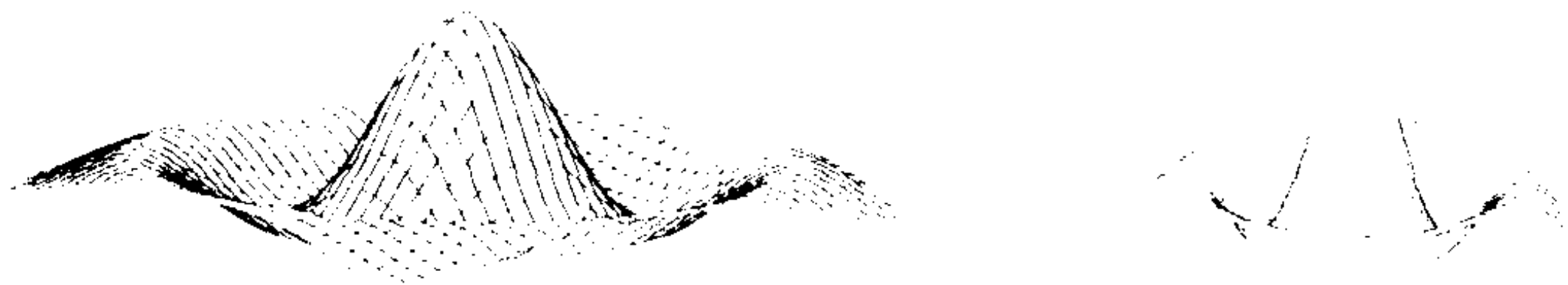
其中竖向放缩因子是可选项, 若不用该项, 则竖向与横向用相同的放缩因子, 此时可保持图形横竖比例不变. 放缩因子可以是负数, 表示向反方向放缩. 插入的图形既可以是整个图形, 也可以是剪裁后的图形. 输入(参见 7-5-2.tex)

```

\begin{center}
\scalebox{0.4}[0.2]{\includegraphics{pic/pic.eps}}\quad
\scalebox{0.2}{\includegraphics{pic/pic.eps}}
\end{center}

```

输出



放缩到指定大小的命令是

`\resizebox{宽度}{高度}{插入的图形}`

其中的{宽度}和{高度}都要指定,但其中一个可以是{!},这表示在一个方向上放缩到指定尺寸时,仍保持图形原有的横竖比例不变.输入(参见7-5-3.tex)

```
\begin{center}
\resizebox{4cm}{25mm}{\includegraphics{pic/pic.eps}}\quad
\resizebox{1.2in}{!}{\includegraphics*[100,430][400,600]{%
pic/pic.eps}}
\end{center}
```

输出为



7.5.6 水平翻转和旋转

将图形水平翻转的命令是

`\reflectbox{插入的图形}`

这实际上是命令`\scalebox{-1}[1]{...}`的缩写.输入(参见7-5-4.tex)

```
\begin{center}
\reflectbox{\includegraphics{pic/panda.eps}}\quad
\reflectbox{\includegraphics*[10,10][30,40]{pic/panda.eps}}
\end{center}
```

输出为(注意原来头朝左,现在头朝右)



将图形旋转的命令是

`\rotatebox{旋转角度}{插入的图形}`

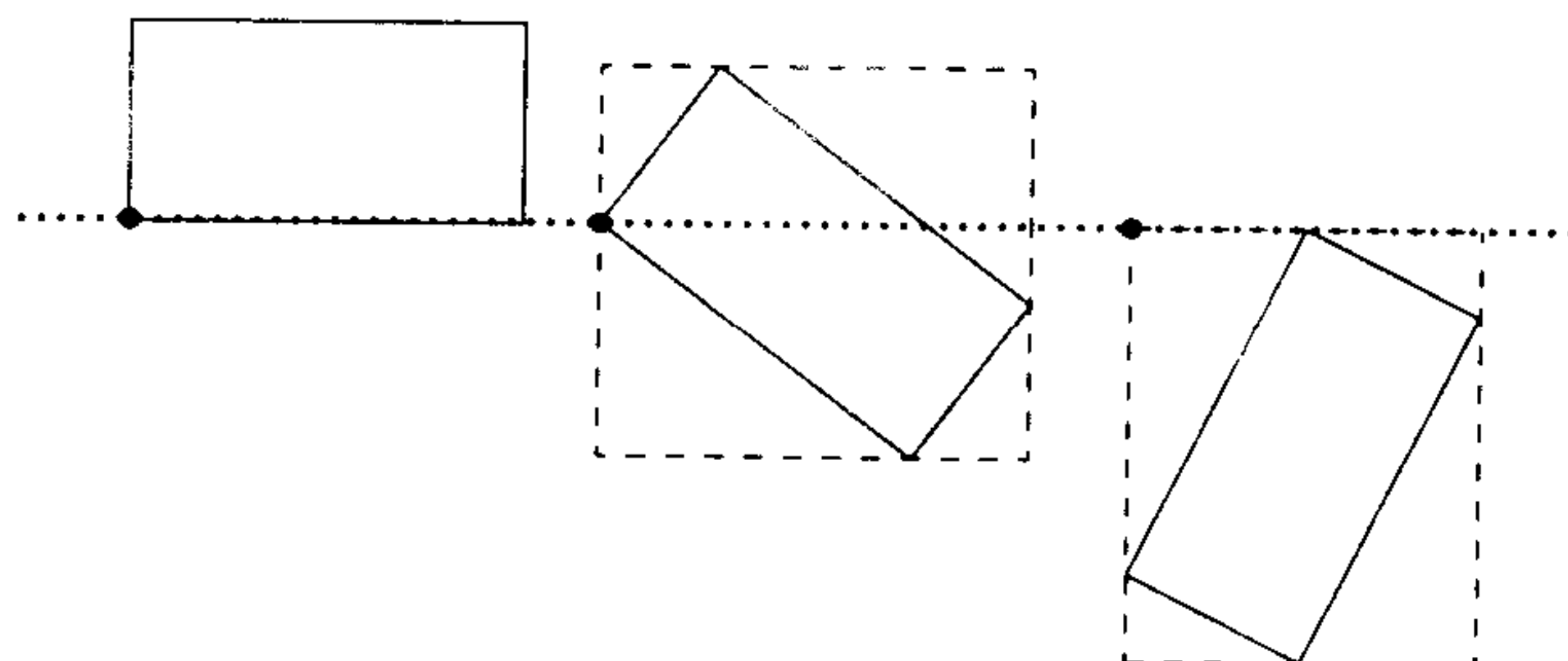
其中旋转角度以度为单位, 逆时针方向为正. 例如输入 (参见 7-5-5.tex)

```
\begin{center}
\includegraphics{pic/panda.eps}
\rotatebox{-30}{\includegraphics{pic/panda.eps}}
\rotatebox{30}{\includegraphics{pic/panda.eps}}
\rotatebox{180}{\reflectbox{\includegraphics{pic/panda.eps}}}
\end{center}
```

输出为



EPS 图形的盒子是其边界盒子, 基准点在左下角. 图形的旋转就是盒子的旋转, 默认值是绕基准点旋转. 旋转后图形区域的外切正矩形是旋转后的图形盒子, 旋转后图形盒子的高度、深度和宽度都会发生变化. 下图是一个盒子顺时针旋转 37° 和 120° 后的情况, 图中圆点是旋转后的基准点, 基准点所在水平线是基线 (图中用虚线表示), 虚线矩形是旋转后的图形盒子.



外部图形也可以放在绘图环境中, 例如输入 (参见 7-5-6.ty)


```

\begin{center}
\begin{picture}(75,42)
\put(0,30){这是一幅}
\put(18,0){\resizebox{!}{4cm}{\includegraphics{photo.eps}}}
\put(60,10){风景照片}
\end{picture}
\end{center}

```

输出为

这是一幅



风景照片

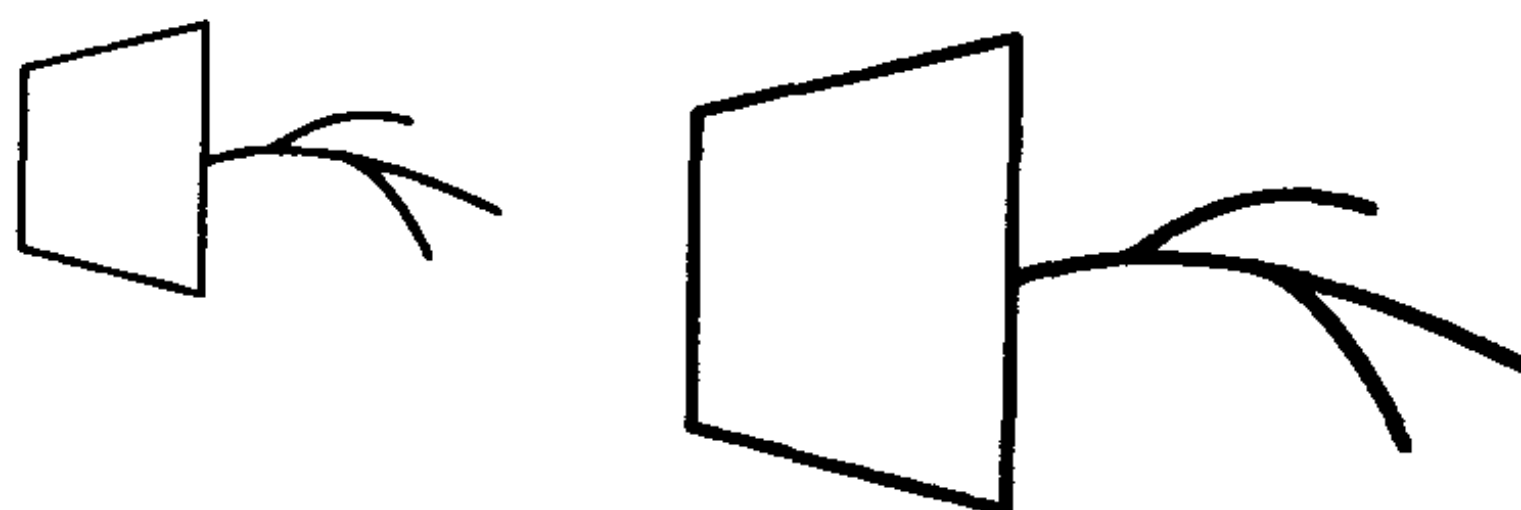
上述命令中的参数{插入的图形}都可以换成普通文本或其他对象, 例如输入 `\rotatebox{-15}{Graphics}` 显示为 *Graphics* (若在预览器上看不到这种效果, 用 `dvips.exe` 转换成 `ps` 文件后再看).

上述的放缩、旋转等命令可以嵌套使用, 但命令的次序对最后结果是有影响的. 下面给出一个例子(参见 `7-5-8.tex`), 其中的对象是第95页画出一盆花. 左边是先顺时针旋转90度, 再放缩到宽度2 cm, 右边是先放缩到宽度2 cm, 再顺时针旋转90度:

```

\begin{center}
\resizebox{2cm}{!}{\rotatebox{-90}{\usebox{\flower}}}
\quad
\rotatebox{-90}{\resizebox{2cm}{!}{\usebox{\flower}}}
\end{center}

```



对显示结果不必惊讶, 看一下这盆花的尺寸就好理解了. 这个图形原始尺寸是宽 12 pt, 高 21 pt, 高度几乎是宽度的一倍. 左边是先旋转 90 度, 旋转后宽度变成了高度的一倍, 然后将旋转后的图形的宽度变为 2 cm, 高度就只有 1 cm 多点; 右边图形是先将宽度变为 2 cm, 高度就几乎为 4 cm 了, 然后再旋转 90 度, 结果的宽度就差不多是 4 cm, 比左边的图形宽了近一倍.

§7.6 浮动表格和图形

`picture` 环境和 `tabular` 环境生成的图形或表格通常是插入在当前位置, 也就是输入的位置. 如果当前页的剩余空间不够, 图表将被移动到下一页, 当前页就会出现很大的空白, 于是需要人工调整图表与前后文本的位置. 一处的调整可能影响后面多处的调整, 文稿有所修改时又要重新调整, 非常不便. L^AT_EX 提供了浮动图表的功能, 遇到图表在当前位置放不下时, 会在一定的原则下自动调整图表的位置, 大大减轻了人们的工作量.

浮动图形和浮动表格环境是

```
\begin{figure}[位置] 图形 \end{figure}
\begin{figure*}[位置] 图形 \end{figure*}
\begin{table}[位置] 图形 \end{table}
\begin{table*}[位置] 图形 \end{table*}
```

对于单列页面格式, 带 * 号的环境与标准环境(无 * 号的坏境)作用相同. 带 * 号的环境只适用于双列页面格式, 它使得图形或表格占据两列, 而不是正常情况的一列.

可选项位置由零个到 4 个字母组成, 这些字母是: h (here, 当前位置), t (top, 页芯顶部), b (bottom, 页芯底部), p (page, 单独一页). 上述字母或字母组合可与符号 ! 连用, 它会去掉对浮动所加的一些限制. 如果不写可选项, 默认是 `tbp`. 本书中带有图号的图形实际是如下输入的:

```
\begin{figure}[!htbp]
\centering
\begin{picture}
.....
\end{picture}
\end{figure}
```

位置选项字母的顺序是无关紧要的, $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X } 2_{\epsilon}$ 总是按 $\text{h} \rightarrow \text{t} \rightarrow \text{b} \rightarrow \text{p}$ 的次序处理选项中的字母.

§7.7 其他绘图软件包介绍

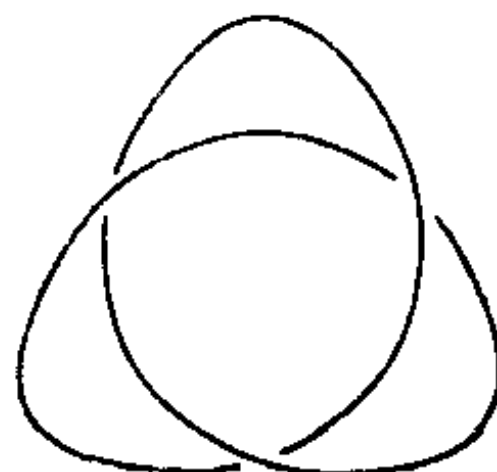
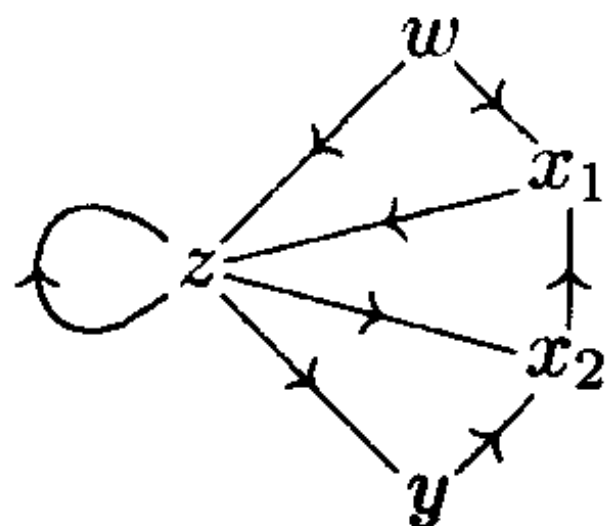
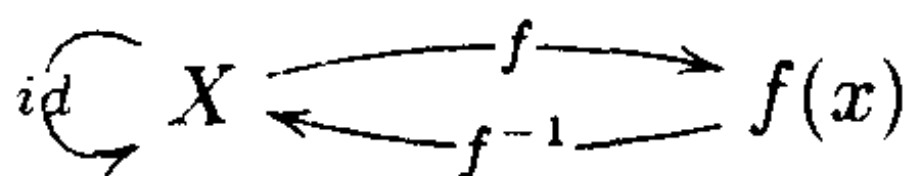
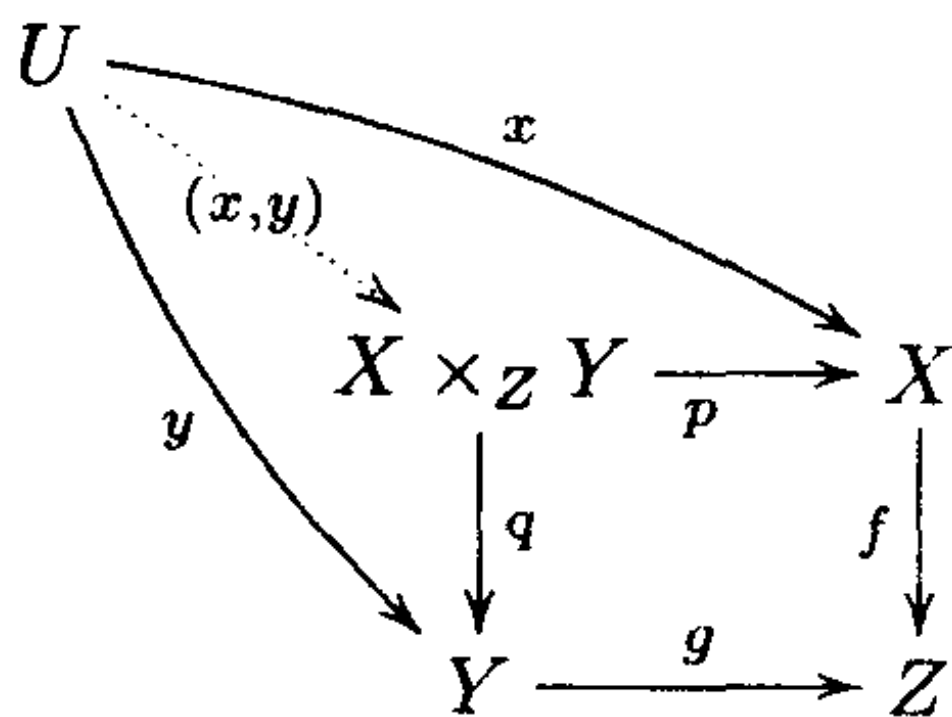
Xy-pic 是 Kristoffer H. Rose 和 Ross Moore 开发的用于绘图, 尤其是交换图的 $\text{T}_{\text{E}}\text{X}$ 宏包, 是一个自由软件. 在 MiKTeX 的基本安装中不包含这个宏包, 读者如想使用 Xy-pic, 可进入

开始 \rightarrow 程序 \rightarrow $\text{MiKTeX} \rightarrow \text{MiKTeX Options}$

选择 'Packages' 页, 在 'Download Site' 栏里输入 $\text{F}:\backslash\text{miktex}\backslash\text{tm}\backslash\text{packages}$ (假定光盘盘号是 F), 在左下角的 MiKTeX Packages 方框里依次通过点击左侧的田符号进入 'Applications \rightarrow Graphics', 点击其中的 'xypic' 条左边的小方框, 使得里面出现一个小勾, 表示选中, 点击右下方的 '应用' 按钮, 就会自动安装这个宏包.

由于 Xy-pic 用 METAFONT 建立自己的字模, 因此它作出的图形不依赖 PostScript. Xy-pic 除了在画交换图, 尤其是含有弧线的交换图方面有它的特色外, 它可以画图论中用到的各种有向或无向图, 以及画扭结图, 这些都是它的特色, 读者可从下面几个实例 (参见 7-7-1.tex) 看到 Xy-pic 的能力. 其中第一个交换图的源程序如下.

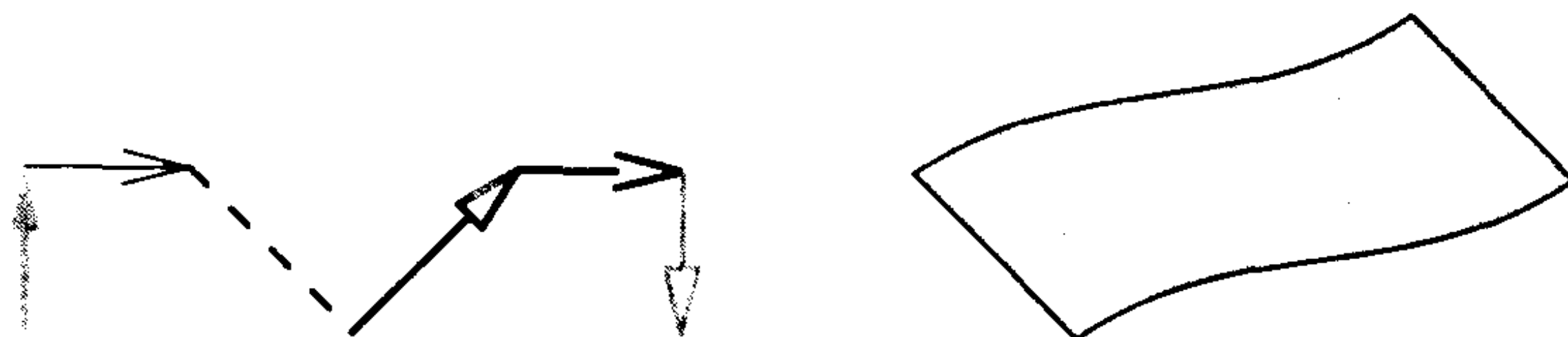
```
\xymatrix{
  U \ar@/_/[ddr]_y \ar@/^/[drr]^x \\
   \ar@{.>}[dr]|-{(x,y)} \quad \backslash \\
  & X \times_Z Y \ar[d]^q \ar[r]_p & X \ar[d]_f \\
  & Y \ar[r]_g & Z}
  & X \ar@<[r]_{f^{-1}} \ar@>[r]^f & f(x)
  & \text{twisted knot diagram}
```



另一个配合 TeX 的绘图宏包是 Peter Kabal 开发的 T_EXdraw. 这也是一个自由软件. 由于 MiKTeX 的基本安装也不包括这个宏包, 读者可使用第 239 页中介绍的安装 Xy-pic 宏包一样的方法安装 T_EXdraw, 只要在最后一步把选中 'xypic' 改成选中 'texdraw' 即可.

T_EXdraw 的输出是含有 PostScript 命令的附加文件, 生成 dvi 文件后要用 dvips 转化成 ps 文件才能在 PostScript 打印机上打印出图像. 不过现在有不少 dvi 驱动软件也能打印 T_EXdraw 作出的图像, 例如 MikTeX、fpTeX 以及 PCTeX32 的预览程序都能显示以及打印这些图像. 由于 PostScript 的强大功能被调动起来, 因此 T_EXdraw 的特色是产生各种形状以及灰度的箭头图, 还能在闭合图形内加阴影, 不过它的曲线图形的功能不多, 除了 Bézier 曲线外, 只能画圆弧和椭圆弧. 下附的两个图显示了 T_EXdraw 的特色, 其源程序如下 (参见 7-7-2.tex).

```
\begin{texdraw}
\drawdim cm
\linewidth 0.06 \setgray 0.6 \arrowheadtype t:F
\avec(0 1)
\linewidth 0.02 \setgray 0 \arrowheadtype t:V
\avec(1 1)
\linewidth 0.03 \lpatt(0.15 0.2) \lvec (2 0)
\linewidth 0.04 \lpatt() \arrowheadtype t:T
\avec(3 1)
\arrowheadtype t:H \avec(4 1)
\setgray 0.4 \arrowheadtype t:W \avec(4 0)
\end{texdraw}
```



功能最强的科学绘图软件可算是 John D. Hobby 开发的 MetaPost 了. 它利用 METAFONT 的基本工具把图像输出为 PostScript 的程序, 因此充分利用了这两者的强大绘图功能. MetaPost 能对点的坐标作线性运算, 能求直线或曲线间的交点, 能画参数曲线, 能对图形作仿射变换 (当然包括旋转与翻转), 能在闭合曲线所围

的区域里画阴影以及着色等. MetaPost 实际上是一种作图的预处理程序, 用户先把要作的图形写成源程序, 储存成以 `mp` 作为扩展名的文件, 然后用 MetaPost 程序编译这个 `mp` 文件, 生成与源文件同名而扩展名则是一个数字的 PostScript 程序, 然后在 TeX 源文件中加入命令读入这个 PostScript 文件, 经 TeX 编译后生成 `dvi` 文件, 再通过 `dvips` 转化为 `ps` 文件, 就可以用具有 PostScript 功能的打印机打印了, 也可以利用 GhostScript 软件在普通打印机上打印. 这就是 MetaPost 的工作流程.

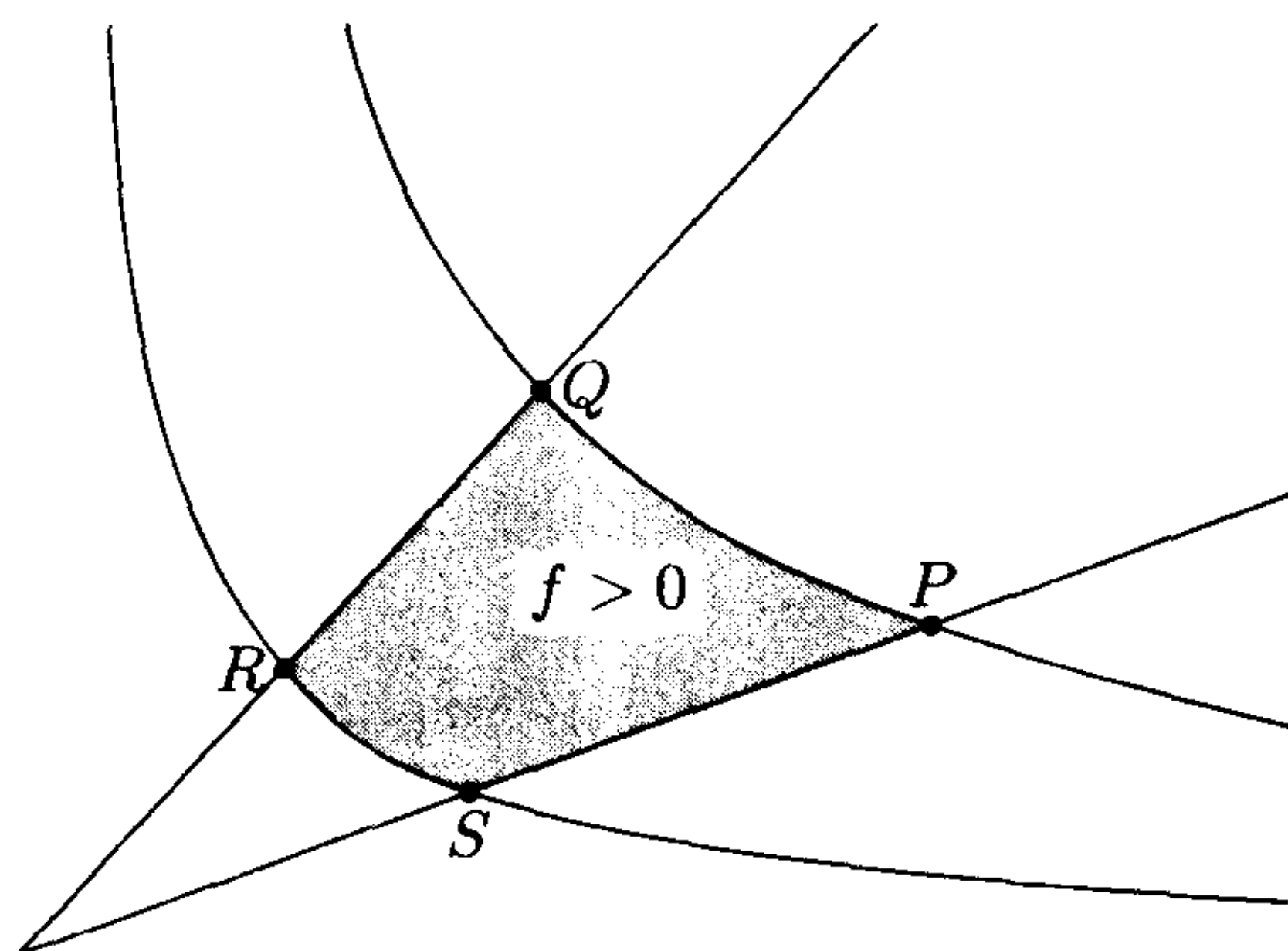
MikTeX 的基本安装中已经包含了 MetaPost 的执行程序, 名为 `mp.exe`, 还可以找到 MetaPost 的完整说明书. 作为样本, 我们给出下面的名为 `fig.mp` 的 MetaPost 源文件.

```
beginfig(1);
h:=2in; w=2.7in;
path p[], q[], pp;
for i=2 upto 4: ii:=i**2;
p[i]=(w/ii,h){1,-ii}...(w/i,h/i)...(w,h/ii){ii,-1};
endfor
q0.5=(0,0)--(w,0.5h);
q1.5=(0,0)--(w/1.5,h);
pp=buildcycle(q0.5,p2,q1.5,p4);
fill pp withcolor .7white;
z0=center pp;
picture lab; lab=thelabel(btex $f>0$ etex, z0);
unfill bbox lab; draw lab;
draw q0.5; draw p2; draw q1.5; draw p4;
dotlabel.top(btex $P$ etex, p2 intersectionpoint q0.5);
dotlabel.rt(btex $Q$ etex, p2 intersectionpoint q1.5);
dotlabel.lft(btex $R$ etex, p4 intersectionpoint q1.5);
dotlabel.bot(btex $S$ etex, p4 intersectionpoint q0.5);
endfig;
```

经 MetaPost 编译后生成名为 `fig.1` 的 PostScript 程序文件, 再在 TeX 源文件中用以下命令把图形嵌入正文 (参见 `7-7-3.tex`):

`$$\includegraphics{fig.1}$$`

打印结果如下图所示.



应该说 MetaPost 的作图功能是很强的, 但是编程有一定技巧, 而且大部分功能也能由 Tydraw 做到, 并且与 Tydraw 比, MetaPost 有三个缺点: 一是处理速度比 Tydraw 慢; 二是要专门生成图形文件, 不像 Tydraw 可以嵌在中文 TeX 源文件内, 一次处理完毕, 而且在 MetaPost 的图形中要加入中文标注也比较复杂; 三是要使用 PostScript 打印机或者借助 GhostScript, 多了一道手续. 所以除了特殊需要外, 使用 Tydraw 作图已经足够了.

第八章 如何使文本竖向对齐

§8.1 使文本居左或居右

类似于居中对齐环境(见第28页), 居左对齐环境

```
\begin{flushleft}  
  第一行\\  
  第二行\\  
  .....  
  第末行\\  
\end{flushleft}
```

使得文本靠左对齐. 对于超长文本自动分行, 不进行断词并且使单词间距相同.
类似于\centering, 在一个分组或环境内部, 可以使用声明

```
\raggedright
```

使后继文本左对齐, 声明的作用到分组或环境结束时为止. 同样可以使用\\强制分行. 这个声明的字面含义是“右边可以不齐”, 实际作用是“左边一定对齐”.

居右对齐环境和居右对齐声明分别是

```
\begin{flushright}  
  第一行\\  
  第二行\\  
  .....  
  第末行\\  
\end{flushright}
```

```
\raggedleft
```

除了使文本右对齐外, 其他与左对齐环境和声明是类似的.

紧跟在对齐环境后的文本如果与环境之间没有用空行分开, 则认为与环境中的文本属于同一个段落, 即此时环境后的第一行没有段首缩进, 但与环境中的文本仍有额外的竖直间隔.

§8.2 引文

在文章中的大段引文常常是另起一段, 并且将段落两边向内缩进一定距离. 下面两个环境都可实现这个要求:

```
\begin{quotation}
  文本
\end{quotation}
```

```
\begin{quote}
  文本
\end{quote}
```

其中文本可以很短, 只有几个字, 也可很长, 甚至包含几个段落.

上述两个环境有如下区别: 第一种环境中的段落象正常段落一样, 有首行缩进(前提是通常的段落是有首行缩进的), 段落之间具有正常的间距. 第二种环境并不自动首行缩进, 而是自动增大了段落间距.

对于上述两种环境, 在显示文本的上方和下方都被自动插入了额外的间隔. 排版西文诗歌、韵文时, 也是需要两边缩进, 所用环境是

```
\begin{verse}
  诗歌、韵文
\end{verse}
```

排版效果类似于 `quote` 环境, 即段落(诗歌的小节)首行不缩进, 段落间隔加大. 但在诗歌环境中对超长的行自动分行时, 分成的几行不是靠左对齐而是左端“悬挂”, 即后面几行比第一行要向右缩进一点距离.

上述引文及诗歌环境可以相互或自我嵌套, 最多嵌套6层. 每个内层环境都比外层环境更向内缩进.

§8.3 抄录

有时我们希望不对某些文稿进行格式处理, 而是按照输入的文本原样输出, 例如原样输出一段缩排的程序, 所有空格和空行也要保持原样, 这可使用下列抄录环境来实现:

```
\begin{verbatim}  
    文本  
\end{verbatim}
```

```
\begin{verbatim*}  
    文本  
\end{verbatim*}
```

其中的文本会用打字机字体按原样输出, 文本内的控制命令也被当成普通字符串原样显示出来. 但要注意抄录环境内不能有汉字.

上述两种抄录环境的区别是显示空格的不同, 不带*号(称标准形式)的环境用空白空格即看不见的空格显示空格, 而带*号的环境则是用␣显示空格.

对于不超过一行的文本, 两个抄录环境可以简化成两个抄录命令:

```
\verb|文本|  
\verb*|文本|
```

其中表示定界符的竖线“|”可以换成除*号以外的任何没有出现在文本内的符号, 左右定界符必须使用同一个符号. 带*号的命令与带*号的环境一样用可见的符号␣显示空格.

需要注意几点: 一是抄录环境和抄录命令都不能用作其他命令的参数; 二是不用抄录环境而用抄录命令时, 文本不能多于一行(在文稿中必须输入在同一行上); 三是键盘上的重音号“`”和单引号“'”在抄录环境或抄录命令中仍是显示为英文单引号“'”和“'”.

§8.4 罗列

8.4.1 三种罗列环境

写文章做报告时, 常常要将文字材料分成几个层次, 按甲乙丙丁一二三四罗列出来. L^AT_EX 提供了三种罗列环境:

```
\begin{itemize}
  罗列条目1
  罗列条目2
  ...
  罗列条目n
\end{itemize}
```

```
\begin{enumerate}
  罗列条目1
  罗列条目2
  ...
  罗列条目n
\end{enumerate}
```

```
\begin{description}
  罗列条目1
  罗列条目2
  ...
  罗列条目n
\end{description}
```

其中所有罗列条目都有如下形式:

```
\item[标签] 条目文本
```

输出时在每个条目的前面显示标签, 标签既可以是符号, 也可以是文字. 若一个条目文本的内容很长, 会自动分行, 分出来的后面几行与第一行的文字对齐, 标签则凸出悬挂于条目的第一行左端, 其中环境 `description` 还会用黑体显示标签 (用汉字作标签时, 不会自动用黑体显示, 可使用命令 “\黑” 指定黑体).

上述3种环境的区别是对可省略的参数 “[标签]” 的处理方式不同.

当不写 [标签] 时, 环境 `itemize` 会自动生成默认的条目标签, 其特点是并列的条目有同样的标签; 环境 `enumerate` 也会自动生成默认的条目标签, 那是自动排序编号的符号; 但环境 `description` 没有默认的标签, 因此在这种罗列环境中不省略 [标签] 项.

上述3种罗列环境可以彼此相互嵌套,每一种环境最多嵌套4层,嵌套时内层的环境会向右整体缩进.前两种环境的默认条目标签与该环境自己的嵌套层数有关.`itemize`环境的4层默认标签分别是 \bullet , $-$, $*$ 和 \cdot . `enumerate`环境第一层默认标签是阿拉伯数字后跟一个圆点句号,第二层是圆括号包围的小写拉丁字母,第三层是小写罗马数字后跟圆点句号,第四层是大写拉丁字母后跟圆点句号,标签是自动顺序编号的,如果某个罗列条目指定了标签,则该项不参加自动编号.

8.4.2 改变默认的罗列条目标签

改变某一项的默认标签是很简单的,只要在输入`\item[标签]`条目文本时,不省略标签就行了.若要改变所有项的默认标签,需先了解 \LaTeX 的一些内部命令.

`itemize`环境的4层默认标签分别保存在下列4个命令中:

```
\labelitemi    \labelitemii    \labelitemiii
\labelitemiv
```

容易猜出命令名字中的`i`,`ii`,`iii`,`iv`分别指的是四个层次中的一层.可以用重定义命令改变每层的默认标签,例如将第二层的默认标签 $-$ 改为 $+$,可使用命令:

```
\renewcommand{\labelitemii}{+}
```

类似地,在`enumerate`环境中保存每层标签的命令是:

```
\labelenumi    \labelenumii    \labelenumiii
\labelenumiv
```

但因标签是顺次编号的,所以对应于每层还有一个计数器,计数器名字分别是(没有倒斜线!):

```
enumi    enumii    enumiii    enumiv
```

计数器的值可以用下列5种命令显示成不同形式:

```
\arabic    \roman    \Roman    \alph    \Alph
```

从名字上就可知道它们分别表示阿拉伯数字、小写罗马数字、大写罗马数字、小写拉丁字母和大写拉丁字母。

如果要把 `enumerate` 环境第一层的标签改为大写字母 A、B、C 等, 把第二层标签改为阿拉伯数字, 但还要显示出它所在的上一层的序号, 即显示的标签形如 A.1、A.2、B.1 等, 第三层不变, 第四层是用小于号和大于号括起来的小写罗马数字, 可如下使用重定义命令:

```
\renewcommand{\labelenumi}{\Alph{enumi}}
\renewcommand{\labelenumii}{\Alph{enumi}.\arabic{enumii}}
\renewcommand{\labelenumiv}{\scriptsize\romannumeral{enumiv}}
```

如果把上述重定义命令放在导言区, 则新的默认标签对整个文档起作用, 否则就只在它们所处的环境或分组内起作用。

下面是一个嵌套例子, 没有重定义默认标签, 输入的原稿如下. 输入时使用了缩排格式, 便于检查错误。

```
\begin{itemize}
\item 这是 itemize 环境的第一层, 标签是黑圆点.
\begin{enumerate}
\item 虽然是总的第二层, 但这是 enumerate 环境的第一层.
\begin{enumerate}
\item 这是 enumerate 环境的第二层.
\item 条目标签与总的嵌套层数无关,
      只取决于它在所属的罗列环境中的层数.
\end{enumerate}
\item 回到了 enumerate 环境第一层, 是这个环境的第二个条目.
\end{enumerate}
\item 回到了 itemize 环境, 条目标签又变成了黑圆点.
\end{itemize}
```

输出结果如下:

- 这是 `itemize` 环境的第一层, 标签是黑圆点.
- 1. 虽然是总的第二层, 但这是 `enumerate` 环境的第一层.
 - (a) 这是 `enumerate` 环境的第二层.
 - (b) 条目标签与总的嵌套层数无关, 只取决于它在所属的罗列环境中的层数.

2. 回到了 `enumerate` 环境第一层, 是这个环境的第二个条目.
- 回到了 `itemize` 环境, 条目标签又变成了黑圆点.

§8.5 广义罗列环境

上述三种罗列环境以及其它一些列表都可以通过一个非常一般的环境构造出来. 标签的类型与宽度, 缩进的距离, 项目、段落、标签等相互之间的距离都可以整体或部分的调整 and 设置. 这就是广义罗列环境 `list`:

```
\begin{list}{标准标签}{罗列声明}
\item[标签] ...
\item[标签] ...
\end{list}
```

罗列的每一项都由 `\item` 引导以便生成相应的标签. 若 `\item` 不带标签可选项, 则使用标准标签的内容作为该项的标签.

有很多参数控制广义罗列环境的排版效果, 用户可以使用默认值, 也可在罗列声明中将参数设置为所期望的值.

8.5.1 标准标签

标准标签就是默认标签, 当广义罗列环境不带可选项标签时, 就使用标准标签作为它的标签.

如果希望标准标签是自动递增的数字, 就必须使用命令

```
\newcounter{名称}
```

建立一个新的计数器, 该命令应放在第一次使用它的计数值之前. 与其他计数器相同, 可以用不同的形式显示计数器的值 (见第 124 页).

若在标准标签中使用一个计数器, 必须在罗列声明中包含命令

```
\usecounter{名称}
```

8.5.2 广义罗列环境的样式参数

有很多样式参数控制广义罗列环境的排版效果, 它们都有默认值, 用户可以在罗列声明中用 `\setlength` 命令改变这些值. 由于每个参数在每一层都有预设的默认值, 而这些值只有在罗列声明中才可以被覆盖, 所以在广义罗列环境之外通常不能重设参数的值.

下面介绍广义罗列环境的样式参数, 需参考图 8.1 以便于理解.

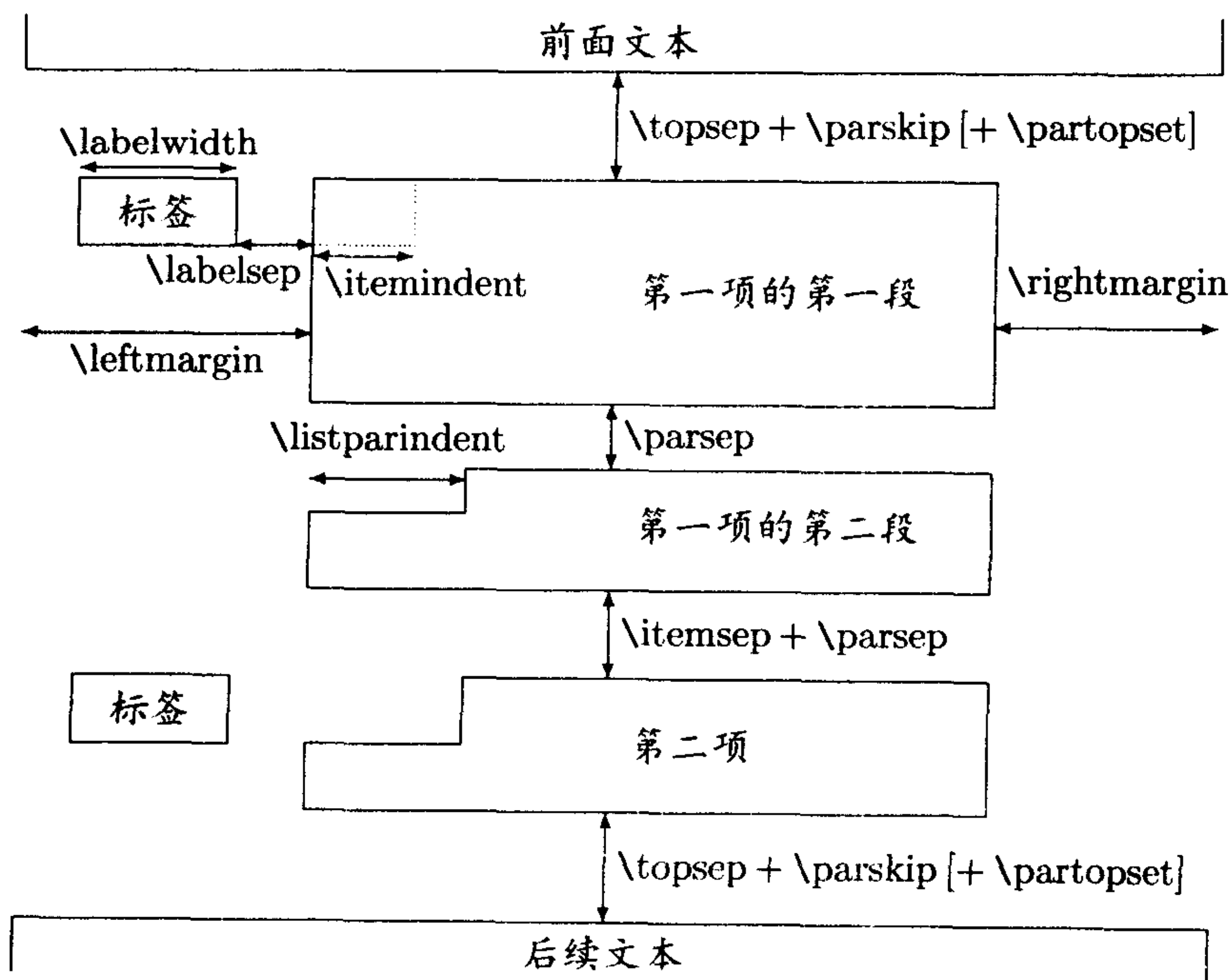


图 8.1 广义罗列环境参数示意图

`\topsep` 是一个竖直间距, 与 `\parskip` 加在一起插入到罗列环境与包围它的外部文本 (图中显示的前面文本和后续文本) 之间. 其值需在每个罗列环境中设置, 即使罗列环境嵌套, 也需在每个环境的罗列声明中改变默认值, 不能在罗列环境之外作全局设定.

`\partopsep` 当在广义罗列环境的前面或后面有空行时, 就会把这个参数的值与上一条提到的两个参数的值加起来, 插入到该环境与外部的文本行之间.

`\parsep` 相当于普通文本的 `\parskip` 参数, 但用在广义罗列环境内, 控制在同一个 `\item` 项目中, 各个段落之间的竖直间距. 需在每个环境的罗列声明

中改变默认值, 不能在罗列环境之外作全局设定.

`\itemsep` 是一个竖直间距, 与 `\parsep` 加在一起插入到相邻两个 `\item` 项目之间.

`\leftmargin` 是当前罗列环境内文本的左边界与外部环境左边界之间的距离.

`\rightmargin` 是当前罗列环境内文本的右边界与外部环境右边界之间的距离.

默认值为 0pt, 只能在罗列声明中改变.

`\listparindent` 相当于普通文本的 `\parindent` 参数, 是 `\item` 项中每一个段落的第一行相对于罗列文本左边界缩进的距离. 默认值为 0pt, 只能在罗列声明中改变.

`\labelwidth` 是为标签保留的盒子的宽度. 在这个盒子中, 标签靠右对齐在这个盒子的右边线上. 但若标签的长度超过了 `\labelwidth` 的值, 这个盒子会向右伸展以拟合标签的长度, 此时罗列文本将向右缩进, 在标签与罗列文本首行之间仍有 `\labelsep` 的间隔. 可以为这个参数设置一个全局性的值以适用于所有的罗列层次.

`\labelsep` 是标签盒子与罗列文本之间的距离. 可以在外部为其设置一个全局性的值, 但只对嵌套罗列的第一层起作用.

`\itemindent` 是标签盒子与罗列文本第一行向右缩进的距离. 通常设为 0pt, 因此没有这种缩进. 只能在罗列声明中改变其值. 图中用虚线表示了文本缩进的这个距离, 没有反映出标签也应缩进同样的距离. 通常罗列的每一项都是悬挂效果(首行突出), 如果要改变成首行缩进(连标签一起缩进), 就需要使用这个参数了.

利用上述参数, 可以随心所欲地设计各式罗列(列表)格式, 例如想要罗列的各项之间以及同一项内的各段之间的间距都与正常文本的段落间距相同, 各项标签左对齐, 标签含有序号, 左右相对于正文缩进 2em, 形如

问题 9: A、B、C 三射手各自独立地向同一目标进行一次射击, 已知 A、B、C 射中目标的概率分别为 $\frac{1}{3}$ 、 $\frac{1}{2}$ 、 $\frac{1}{4}$, 则至少有一人射中目标的概率为 _____

问题 10: 设 $\xi \sim N(\mu, \sigma^2)$, 则 $D\xi =$ _____

可以如下输入:

```
\newcommand{\TKX}[1]{\,\,\,\rule[-3pt]{#1mm}{0.5pt}}
\newcounter{timu}
\begin{list}
{\bfseries\upshape\黑问题\,\arabic{timu}:\hfill}
```



```

{\setlength{\parsep}{\parskip}
 \setlength{\itemsep}{0ex plus0.1ex}
 \setlength{\rightmargin}{2em}
 \setlength{\leftmargin}{6.2em}
 \setlength{\labelwidth}{4em}
 \setlength{\labelsep}{0.2em}
 \usecounter{timu} \setcounter{timu}{8}
 \slshape
}
\item A、B、C 三射手各自独立地... 概率为\TKX{12}
\item 设..., 则  $D\$ \xi = \$$ \TKX{12}
\end{list}

```

根据罗列参数的值, 容易算出 $6.2\text{em} - 4\text{em} - 0.2\text{em} = 2\text{em}$ 就是标签左边界与页面左边界的距离.

注意语句 `\usecounter{timu}\setcounter{timu}{8}` 的顺序, 如果次序颠倒, 则第一个标签序号是 1.

在标准标签和罗列声明中可以分别指定西文字体属性.

8.5.3 平凡罗列环境

在 L^AT_EX 中还有一个平凡罗列环境:

```
\begin{trivlist} 文本 \end{trivlist}
```

它相当于一个广义罗列环境, 但没有标准标签和罗列声明参数, 其标签是空的, `\leftmargin`、`\labelwidth` 和 `\itemindent` 都赋值为 0pt, 而 `\listparindent` 等于 `\parindent`, `\parsep` 等于 `\parskip`.

L^AT_EX 用这一环境生成其它一些环境. 例如当调用 `center` 环境时, 内部实际上是调用

```

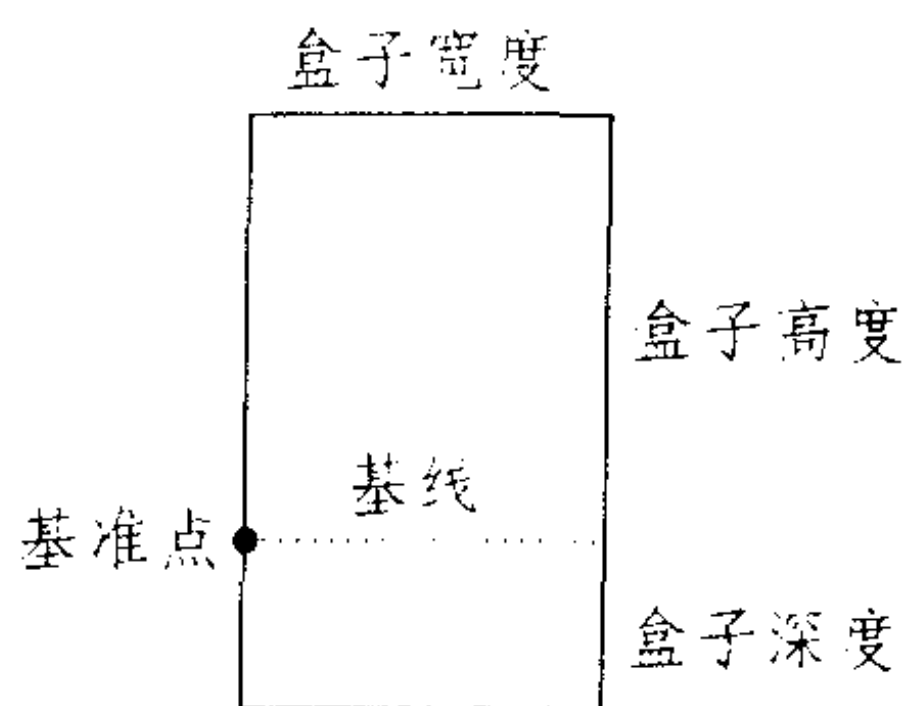
\begin{triivlist}
\centering \item[] 文本
\end{trivlist}

```

环境 `flushleft` 和 `flushright` 也是类似定义的.

§8.6 盒子

在 TeX 中一切都是盒子, 单个字符是盒子, 若干字符盒子排成一行构成大一点的行盒子, 若干行盒子堆叠成段落盒子或页盒子. TeX 排版的过程实际就是在构造盒子和排列堆叠盒子. 盒子之间插入了可伸缩的弹性长度. 除了上述这些隐含的盒子, 还有一些显式生成盒子的命令, 后面会详细介绍. 每种盒子无论是带框的还是无框的, 都占据了一个矩形区域, 如下图所示:



图中的虚线称为盒子的“基线”, 当盒子相互边靠边排列时, 它们的基线总是位于同一条水平线上. 基线的左端点(图中的小黑点)称为盒子的“基准点”或“参考点”. 每个盒子都有3个尺寸: 高度、深度和宽度, 注意盒子高度不是整个矩形的高度而是基线上方的高度. 矩形的高度称为总高度, 等于盒子高度与盒子深度之和. 这些尺寸用下列命令表示:

<code>\width</code>	表示盒子的宽度
<code>\height</code>	表示盒子的高度
<code>\depth</code>	表示盒子的深度
<code>\totalheight</code>	表示盒子的总高度

L^AT_EX 用户可以使用命令构造三种类型的盒子: LR 盒子、段落盒子和标尺盒子. LR (左-右) 盒子中包含的是从左到右的排版对象, 它们排在同一行上, 不能被分割. 段落盒子由竖直堆叠的行组成. 标尺盒子(或称划线盒子)是一个实心矩形, 通常用于画水平或竖直线段.

8.6.1 LR 盒子

下列4个命令都可以创建 LR 盒子:

```
\mbox{单行文本}
\fbbox{单行文本}
\makebox[宽度][位置]{单行文本}
\framebox[宽度][位置]{单行文本}
```

单行文本的含义是它不能被分行,但它内部可以含有作为一个整体的含有多行文本的环境或盒子.

前两条命令生成的 LR 盒子,其宽度等于单行文本排版后的宽度,通常称为自然宽度.这两个命令的区别是上一个命令产生的盒子没有边框,下一个则产生带边框的盒子,把单行文本包围起来.

后两条命令产生的 LR 盒子的宽度由可选项宽度给定.另一个可选项位置指定单行文本在盒子中的排放位置,它可取如下的值:

- l 文本的左端紧靠盒子的左端
- r 文本的右端紧靠盒子的右端
- s 伸展单词间隔使文本撑满盒子
- 缺省 文本的中点对齐在盒子的中点

对于后两个命令,如果两个可选项都缺省,那就等同于前两个命令了.如果省略一个,则只能省略[位置],此时表示盒内文本居中排列.后两个命令的区别也是上一个产生无框盒子,下一个产生有框盒子.

当指定的宽度小于单行文本的自然宽度时,文本就会超出盒子,根据位置的值很容易确定文本是在盒子的左边、右边还是两边突出出来.例如

`\framebox[6em][l]{左对齐时向右突出}` 结果为 左对齐时向右突出

`\framebox[6em][r]{右对齐时向左突出}` 结果为 右对齐时向左突出

`\framebox[6em]{中间对齐时向两边突出}` 结果为 中间对齐时向两边突出

上述例子仅仅为了形象地说明对齐与超出方向的关系,没有多少实用价值.但若改用 `\makebox` 并指定宽度为零,则在 `picture` 环境中就有用武之地了,用它很容易在指定的作图点生成居中对齐或左(右)对齐的文本.用它也可以使两部分文本重叠,例如

`\makebox[0pt][l]{/\backslash}` 结果为 X

命令中的宽度可以利用盒子的自然尺寸来定义,所谓自然尺寸就是相当于用 `\mbox` 产生的盒子的尺寸,不是人为指定的尺寸.例如

`\framebox[2\width][r]{两倍宽度}` 结果为 两倍宽度

其中的 `\width` 代表的就是文本的自然宽度,前面的数字 2 是乘法因子.

表示盒子尺寸的命令可用在 LR 盒子的宽度选项中或用在段落盒子的高度选项中,不要用在其他地方以免产生出错信息.

盒子框线与内部文本之间有一定间隔,命令 `\fboxsep` 保存这个间隔值.盒子的框线有一定的粗细,其值保存在命令 `\fboxrule` 中.可按需要修改这两个值:

```
\setlength{\fboxsep}{长度}
\setlength{\fboxrule}{长度}
```

上述设置长度的命令放在导言区时, 对整个文档起作用, 若放在某个环境或分组内, 则仅在局部范围起作用. 在前面演示文本超出盒子的例子里, 上述两个值分别设置为 1pt 和 0.2pt, 其它例子使用的是默认值 (5pt 和 1pt).

如果某些文本在文档中多次出现, 可以用一个盒子把它保存起来, 以后使用就方便了. 首先用命令

```
\newsavebox{\盒子名}
```

创建一个名为 \盒子名的盒子. 注意名字不能与任何已有的命令重名, 并要符合命令的命名规则 (斜线后面只能跟字母). 然后就可用命令

```
\sbox{\盒子名}{文本}
\savebox{\盒子名}[宽度][位置]{文本}
```

把文本的内容保存在盒子中. 各个参数的意义与建立 LR 盒子的命令参数相同. 以后随时可用命令

```
\usebox{\盒子名}
```

把保存在盒子中的文本当作一个整体插入到文档中. 这样的盒子是无框的, 若想加框, 可写成 \fbox{\usebox{\盒子名}}.

L^AT_EX 2_ε 还提供了如下环境用于保存 LR 盒子:

```
\begin{lrbox}{\盒子名}
  文本
\end{lrbox}
```

它的功能与 \sbox 类似, 当保存大段的文本时, 使用这个环境较好, 以后引用时也是用命令 \usebox{\盒子名}. 但不要忘记, 仍要事先创建这个盒子名.

8.6.2 LR 盒子的升降

使用下述命令可以使 LR 盒子的文本竖直移位:

```
\raisebox{升高量}[盒高][盒深]{文本}
```


升高量为正时是升高, 为负时是下降. 该命令生成一个无框的 LR 盒子, 盒子的基线位于当前行的基线上, 但盒内的文本的基线相对于盒子基线竖直移动了升高量的距离.

当没有可选项 [盒高][盒深] 时, 盒子的尺寸由文本和升高量决定: 在保持盒子的基线位置不变的前提下, 自动调整盒子的大小, 使得升降后的文本刚好不会超出盒子.

当有 [盒高][盒深] 时, 盒子尺寸由给定的盒高与盒深的值确定, 此时若升高量过大, 文本会超出盒子的范围. 当盒高或盒深的值是负数时, 被当成零处理.

下面是一个例子, 为了看出盒子的范围, 给一些盒子加了边框:

...ABCD...EF...GH...IJ...KL...MN...

盒外的虚线点表示当前行的基线, 也是每个盒子的基线. 盒内的虚线点是盒内文本的基线. 其中

- 第一个盒子是正常的 LR 盒子 (没有升降),
- 第二个盒子是 `\raisebox{10pt}{D...E}`,
- 第三个盒子是 `\raisebox{4pt}[6pt][6pt]{H...I}`,
- 第四个盒子是 `\raisebox{-10pt}{L...M}`.

位于同一行的盒子的基线处于同一水平线上, 这些盒子组成了一个更大的盒子, 大盒子的高度与深度分别等于各个小盒子的高度和深度的最大值. 当升高或降低的盒子使用了可选项参数, 但给定的高度或深度较小而升高值较大时, 会使升高或降低的字符超出整个行盒子的范围, 从而与相邻行重叠, 这一点是需要注意的. 高度或深度为零的升降盒子对排版一些特殊的表格和矩阵是必不可少的.

8.6.3 标尺盒子

标尺盒子就是一个实心矩形, 命令

`\rule[升高量]{水平宽度}{竖直高度}`

生成一个具有指定宽度和高度实心盒子. 可选项升高量指定了标尺盒子底部相对于当前行基线向上移动的距离, 若值是负的, 则向下移动, 若不使用这个可选项, 则标尺盒子底部就放在当前行的基线上. 下面是两个例子:

`\rule{5pt}{10pt}`

`\rule[4pt]{20mm}{1pt}`

宽度为零高度不为零的盒子是看不见的, 这是一个没有宽度的柱子, 可起“支撑”作用, 比较下面两例的结果:

`\fbox{没有支撑}` 没有支撑

`\fbox{使用支撑\rule[-3mm]{0pt}{8mm}}` 使用支撑

8.6.4 子段盒子与小页环境

用户有两种手段可以显式地把整个段落放到一个竖直盒子中. 一种是使用子段盒子命令:

`\parbox[位置]{宽度}{文本}`

另一种是使用小页环境:

`\begin{minipage}[位置]{宽度}`
 文本
`\end{minipage}`

两者都生成一个具有给定宽度的竖直盒子. L^AT_EX 就象处理通常段落一样, 将盒内的文本分行, 然后堆叠成段 (这是称为“竖直”盒子的缘由). 可省略的参数位置可取如下值:

- b 盒子底行文本基线与盒子外面基线对齐
- t 盒内顶行文本基线与盒子外面基线对齐

当不使用可选项位置时, 盒子的中部与盒外基线对齐.

在排版时, T_EX 对待盒子象对待单个字符一样, 把盒子看成是不可分割的一个大“字符”. 盒子与盒子或盒子与普通字符水平排列时, 基线对齐在同一条水平线上, 上下排列时, 基准点对齐在同一条竖直线上. 上述建立竖直盒子时的位置参数, 实际上是确定竖直盒子的基线位置和基准点位置, 基准点就是基线与盒子左边界的交点.

使用上述命令或环境建立竖直盒子时, 没有指定总高度, 生成的盒子总高度就是所含内容占据的自然总高度.

在 L^AT_EX 中可以指定竖直盒子的总高度, 此时还应指定盒中的内容在盒内摆放的位置, 如靠上、靠下、居中或撑满等, 这就需要增加两个参数, 所以 `\parbox` 命令和 `minipage` 环境更一般的形式是:

```
\parbox[位置][总高度][内部位置]{宽度}{文本}
```

```
\begin{minipage}[位置][总高度][内部位置]{宽度}
  文本
\end{minipage}
```

其中可选项 内部位置 可取如下值:

- t 把文本推向盒子的顶部
- b 把文本推向盒子的底部
- c 使文本竖直居中
- s 伸展行间隔使文本充满整个盒子(应有弹性长度)

总高度可以是数字带一种长度单位,也可以象建立 LR 盒子时的参数一样,使用 `\height`、`\depth`、`width` 和 `\totalheight` 参数. 这些参数相当于是 不使用可选项 [总高度][内部位置] 时, 生成的竖直盒子的相应尺寸.

应同时使用或同时不使用可选项 [总高度][内部位置], 以避免意外的错误.

§8.7 制表位的高级技巧

制表位从左到右顺序编号, 最左边是零号制表位. 在 `tabbing` 环境中, 每一行都是默认从零号制表位开始. 但若把 `\+` 放在一行的开头或末尾, 那么后继行的开始位置就都右移一个制表位. 如果用 `\+\+` 开始或结束一行, 后继行开始位置就都右移两个制表位, 依此类推. 一行设置过几个制表位, 前面最多就可以使用几个 `\+`.

命令 `\-` 具有相反的效果, 每用一次 `\-`, 就抵消一个 `\+` 的作用, 也就是说, 当用 `\-` 开始或结束一行时, 后继行的开始位置就都向左移动一个制表位. 注意在任何位置 `\-` 的总数不能超过 `\+` 的总数.

一行上所有制表位的位置构成一个“制表位表”. 为了清除制表位表以便重新设置制表位, 可以在一行开始时使用命令 `\pushtabs`, 它将现有的制表位表保存到堆栈中以备后用, 然后清除所有的制表位, 于是就可以重新设置制表位了. 在一行开始使用命令 `\poptabs`, 它也是清除当前的各个制表位, 但同时把最近保存在堆栈中的制表位表取出来, 恢复表中的各个制表位. 可以多次使用 `\pushtabs`, 保存不同行上的制表位表. 也可以多次分别地或连续地使用 `\poptabs`, 恢复以前某一次保存的制表位表.

注意, 在任何时候, `\poptabs` 的数目都不能超过已有的 `\pushtabs` 的数目, 并且在 `tabbing` 环境结束时, 要保证 `\poptabs` 与 `\pushtabs` 的数目必须相同. 此外, `\poptabs` 并不保存或取消 `\+` 和 `\-`, 因此不影响任何已有的 `\+` 和 `\-` 的作用.

通常情况下, 在 `tabbing` 环境中, 位于相邻制表位之间的文本, 都是左端对齐在左边一个制表位上 (下称当前制表位). 如果输入的文本形如

左边文本\'右边文本

则是 `\'` 对应的位置对齐在当前制表位上 (该符号本身不显示), 严格讲是右边文本左对齐在当前制表位处, 左边文本右对齐当前制表位, 但左边文本与制表位之间还有一个小小的间隔, 其值由参数 `\tabbingsep` 确定. 用户可用命令 `\setlength` 改变这个参数的值. 可以看出左边文本实际上是位于前面一列的区域内, 因此要当心不要与前一列文本重叠.

命令

`\`文本`

使文本右对齐于当前版面的右边界. 在文本与该行换行命令 `\\` 之间不能再出现 `\>` 或 `\=` 命令.

在 `\tabbing` 环境外面, `\=`、`\'` 和 `\`` 都是重音符号, 如果在该环境内需要这些重音符号, 就要把 `\` 改为 `\a`. 例如若在 `tabbing` 环境生成字符 `ó`、`ò` 或 `õ`, 就必须输入成 `\a'ó`、`\a`ó` 或 `\a=ó`. 命令 `\-` 在 `tabbing` 环境之外是建议断词标志, 但该环境中不允许断行 (只能用 `\\` 换行), 所以就不需要为它定义替代形式了.

§8.8 表格的高级技巧

8.8.1 更多的表格参数

在基础篇中已对表格的常用参数作了介绍 (参见第32页), 这里再补充介绍一些更深入的参数.

构造表格的环境有两个:

```
\begin{tabular}[竖直位置]{列格式}
.....
\end{tabular}
```

```

\begin{tabular*}{整表宽度}[竖直位置]{列格式}
.....
\end{tabular*}

```

其中带*号的环境指定了表格的总宽度, 为达到这个宽度必须在列的间隔中含有弹性长度, 为此可在列格式的某些地方插入命令`@{\extracolsep\fill}` (见下面介绍), 以使后面的列间距可以伸展, 保证表格能达到指定的宽度. 对于不带*号的环境, 表格宽度取决于各列内容的宽度.

在基础篇中已经介绍过竖直位置参数`t`与`b`, 以及列格式参数`l`, `c`, `r`, `|`, `||`, 下面再补充一些列格式的参数:

`p{宽度}` 指定对应的列的宽度, 列的内容过多时会自动分行, 顶行与其它列对齐.

相当于使用了命令`\parbox[t]{宽度}{列文本}`处理该列文本;

`*{数}{格式}` 这是几个相同格式的缩写, 例如可以将格式串`|r||r||r||`缩写成`l*{3}{r||}`或`*{3}{|r||}`.

对应于边界或列间距的格式项还有:

`@{文本}` 称为@-表达式, 在它对应的位置(表格的边界或是两列之间)“吃掉”了原有的列间隔空白, 改为插入指定的文本.

若无文本, 只写`@{}`, 就相当于取消了对应位置的列间隔. 在表格的边界处, 若不画竖线, 则边界处就有宽度为列间距一半的空白. 若想删除这种空白, 可在列格式的开始和结束处写上格式项`@{}`.

如果在列之间插入的指定文本与相邻列之间需要一些空白, 那么必须在@-表达式的文本中包含设置空白的命令, 如`\hspace{...}`. 如果不想在两列之间插入文本, 只想改变间距使其与默认间距不同, 只要写成`@{\hspace{宽度}}`就行了.

如果在@-表达式中含有`\extracolsep{宽度}`, 则会使其后所有的列间距都增加指定的额外宽度, 直到遇到下一个同样命令为止. 这种额外的间距不会被后面的@-表达式吃掉.

控制表格样式的参数有:

<code>\tabcolsep</code>	<code>tabular</code> 和 <code>tabular*</code> 表格环境列间距的一半
<code>\arraycolsep</code>	<code>array</code> 表格环境列间距的一半
<code>\arrayrulewidth</code>	各类表格中画线(包括横线和竖线)的粗细

`\doublerulesep` 双线之间的距离

`\arraystretch` 各类表格中的行距伸缩因子

这些参数都有预定值, 符合大多数人的要求. 用户有特殊要求时, 可以改变预定值. 例如

```
\setlength{\tabcolsep}{2pt}
```

```
\renewcommand{\arraystretch}{1.2}
```

设置了两列之间的间距为 4pt, 将表格行的行距增大了 20%.

上面提到的 array 表格环境通常称为阵列环境, 格式为

```
\begin{array}[竖直位置]{列格式}
  第一行\\
  第二行\\
  ..... \\
  第末行
\end{array}
```

其中参数的意义与 `tabular` 的相同, 只是这个环境只能用于数学模式, 大多用于排版矩阵或方程组, 详见第四章.

8.8.2 几个表格样例

例 1: 下面是一个行高和“列数”都有变化的表格:

Font Size		
Huge 25	huge 20	
LARGE 17	Large 14	large 12
normalsize 10	small 9	footnotesize 8
scriptsize 7	tiny 5	

这个表格是如下输入的:

```
\newcommand{\ZZ}[2]{\rule[#1]{0pt}{#2}}
```

```
\newcommand{\MC}[3]{\multicolumn{#1}{#2}{#3}}
```

```
\begin{center}\begin{tabular}{|c|c|c|}
```



```

\MC{3}{c}{\textbf{Font Size}}\|[5pt] \hline
\MC{3}{|c|}{\ZZ{-8pt}{30pt}\hfill\Huge Huge 25\hfill
\|vline\hfill\huge huge 20\hfill\mbox{}}\| \hline
\ZZ{-6pt}{22pt}\LARGE LARGE 17 & \Large Large 14
& \large large 12\| \hline
\normalsize normalsize 10 & \small small 9
& \footnotesize footnotesize 8\| \hline
\scriptsize scriptsize 7 & \tiny tiny 5 & \hline
\end{tabular}\end{center}

```

在输入这个表格之前定义了两个新的命令,这主要是为了简化输入工作.对于频繁使用的较长命令,定义一个较短的命令,不仅减少了输入量,也减少了输入出错的机会.对于这个例子,要特别注意3点:

第一,为了使标题与表格固连在一起,永远不被分在两页,可以把标题当成是表格的一行,但不使用表格的列格式,这时就要用\MC命令了,它是\multicolumn命令的简化写法.此外还用了\|[5pt]命令增加与下面行的间隔.

第二,有最大文字的那行看起来只有两列,也不符合本表格的列格式,所以也要用\multicolumn命令.该行还用了\|vline打印一段竖线,并使用了可以无限伸长的弹性长度\hfill,使竖线两边的文字具有居中的效果.

第三,在有很大文字的行使用了“看不见的支柱”——新定义的\ZZ命令,以撑高表格高度,使表格线与文字之间有一定的间隔.

例2: 下面表格用到了列格式参数p{宽度}和命令\cline与\raisebox.另外为了简化输入,又定义了几个命令:

```

\newsavebox{\fk}\newsavebox{\kk}
\setlength{\fboxsep}{0pt}
\setlength{\fboxrule}{0.3pt}
\sbox{\fk}{\framebox[3mm]{\rule[-2pt]{0pt}{3mm}}}
\sbox{\kk}{\usebox{\fk}\usebox{\fk}}

```

姓名		性别		贴照片处
生日	□□□□ 年 □□ 月 □□ 日			
住址				
通讯	电话:			
	E-mail:			
备注				

这个表格是如下输入的:

```
{
\newsavebox{\fk}\newsavebox{\kk}
\setlength{\fboxsep}{0pt}
\setlength{\fboxrule}{0.3pt}
\sbox{\fk}{\framebox[3mm]{\rule[-2pt]{0pt}{3mm}}}
\sbox{\kk}{\usebox{\fk}\usebox{\fk}}
\begin{center}
\begin{tabular}{|c|l|p{20mm}|}
\hline
姓名 & \hspace{\stretch{3.5}}\vline\,\,\,性别\,\,\,\vline
& \hspace*{\fill} & \\
\cline{1-2}
生日 & \usebox{\kk}\usebox{\kk}\,\,\,年\,\,\,%
& \usebox{\kk}\,\,\,月\,\,\,\usebox{\kk}\,\,\,日 & \\
\cline{1-2}
住址 & \hfill贴照片处\hfill{} & \\
\cline{1-2}
{} & 电话: & \\
\cline{2-2}
& \raisebox{1.6ex}[0pt]{通讯} & E-mail: & \\
\hline
备注 & \MC{2}{c|}{} & \\
\hline
\end{tabular}
\end{center}
}
```

注意输入表中“通讯”两字时, 使用了上升的盒子, 抬高了这两个字的位置. 这里升高盒子的可选项[0pt]决不可省, 这是因为该项表示盒子高度为零, 盒内的文本虽然升高了, 但因盒高为零, 所以不影响当前行的高度. 如果省略了这个盒高选项, 那末随着盒内文本的升高, 当前行的高度也随着增大, 就与其它行的高度不同了. 想一想, 如果将“贴照片处”放在“生日”行输入, 但仍要显示在上下居中的位置, 应如何输入? 注意, 如果设置升降盒子的深度为零, 不能只写一个可选项[0pt], 这是因为当只有一个可选项时, 默认它代表盒子高度而不是深度.

在表格第二列中, 第二行中的内容最长, 它决定了第二列的宽度. 第一行第二列只有两个汉字和两条竖线, 所以该单元格有很多的空白. 为了让空白按 3.5 : 1 的比例分布在文字两侧, 在“| 性别 |”的右边用了一个不会被吃掉的 `\hfill` “弹簧”, 在左边则放了三个半“弹簧”. 命令

```
\stretch{倍数}
```

生成一个弹性长度, 是 `\fill` 的指定倍数, `\fill` 的基本长度为 0, 可以无限伸展到任意需要的长度.

本例的输入内容被放在一个分组内, 即用花括号括了起来, 这是为了使改变后的参数值 `\fboxsep` 和 `\fboxrule` 仅在这个组内起作用.

§8.9 脚注与边注

写在页面底部的注释称为脚注, 写在页面边上的注释称为边注. 关于脚注的基本知识请参见基础篇的第 36 页.

8.9.1 自动编号的脚注

L^AT_EX 内部有一个脚注计数器 `footnote`, 每次调用 `footnote`, 计数器的值就加 1, 并用阿拉伯数字显示计数器的新值作为脚注标记. 可以使用命令

```
\setcounter{footnote}{初值}
```

重设计数器的初值, 以后的第一个脚注编号等于 初值 + 1. 可以用如下命令改变脚注的数字形式

```
\renewcommand{\thefootnote}{数字样式{footnote}}
```

其中数字样式就是在第 124 页介绍的 5 种计数器显示命令. 对于计数器 `footnote`, 还有一个计数器显示命令 `\fnsymbol`, 它把从 1 到 9 的计数器值显示为下列 9 个符号:

```
*   †   ‡   §   ¶   ||   **   ††   ‡‡
```

由于只有 9 个符号, 所以在调用第十个脚注之前应重设计数器值为 0.

使用下述命令可把脚注标记恢复成阿拉伯数字形式:

```
\renewcommand{\thefootnote}{\arabic{footnote}}
```

在小页环境中, 有独立的脚注计数器, 名为 `mpfootnote`, 与计数器 `footnote` 无关. 小页中脚注的默认标记是小写拉丁字母.

8.9.2 指定编号的脚注

下述命令直接指定脚注编号, 并且不改变脚注计数器的值:

`\footnote[数]{脚注文本}`

其中数是一个整数. 如果指定了脚注计数器的显示形式为`\fnsymbol`, 则数应位于1到9之间.

8.9.3 禁止模式中的脚注

前面说过脚注命令不能位于数学模式、表格、LR 盒子或子段盒子中, 我们把这些模式称为禁止脚注模式. 但可以把脚注命令分成两部分, 在需要注释的地方只写脚注标记, 这可以出现在任何地方, 包括出现在禁止模式内部, 而把脚注文本放在禁止模式之外. 分别写脚注标记和脚注文本的命令为

`\footnotemark`
`\footnotetext{脚注文本}`

或

`\footnotemark[数]`
`\footnotetext[数]{脚注文本}`

第一组命令对应于自动编号的脚注命令, 第二组对应于指定编号的脚注命令. 每组的第一条命令(脚注标记命令)总是紧接在需要注释的文字后面, 排版输出时就在它出现的地方显示脚注标记. 第二条命令(脚注文本命令)则放在禁止模式的外面. 对于同一个脚注, 必须使用同一组的脚注标记命令和脚注文本命令.

需要注意的是, 每遇到一个自动编号的脚注标记命令, 脚注计数器的值就自动加1, 而自动编号的脚注文本命令只使用计数器的值, 不改变计数器的值. 当在禁止模式中使用了多个自动编号的脚注标记命令, 则脚注计数器的值就增加了几次. 等到出了禁止模式, 遇到的第一个自动编号的脚注文本命令使用的计数器的值就与第一个自动编号的脚注标记的值不同了. 解决办法是在第一个自动编号的脚注文本命令之前调整脚注计数器的值:

`\addtocounter{footnote}{负数}`

这里负数的绝对值应是自动编号的脚注标记命令的个数减1. 在第二个及以后的每个自动编号的脚注文本命令之前要使用下述两个命令之一给脚注计数器加1


```
\addtocounter{footnote}{1}
\stepcounter{footnote}
```

8.9.4 边注

在页边上的注释可以使用下列命令生成:

```
\marginpar{边注文本}
```

它把边注文本显示在当前页面的右侧页边上, 边注第一行基线与命令所在行基线齐平. 边注文本通常被放在一个较窄的子段盒子中, 因此要对边注文本分行.

默认边注出现在页面右边. 如果使用了双面选项 `twoside`, 则显示在“外”侧, 即奇数页显示在右页边, 偶数页显示在左页边. 若使用了双列选项 `twocolumn`, 则显示在两列的“外”侧, 即左列的左侧, 右列的右侧. 本页出现的边注是在前面边注 ‘显示在“外”侧边’ 语句之后插入命令 `\marginpar{\这是\\一个\\边注}` 得到的.

`\marginpar` 把边注文本作左对齐处理, 这当边注位于页面右边时没有问题, 当位于左边时就会离正文过远, 此时应在边注文本前加上命令 `\flushright`.

如果边注带有方向, 例如是一个指向页芯的箭头, 那么位于左边或右边时, 边注的内容应有所不同, 此时应使用命令

```
\marginpar[\flushright左边注文本]{右边注文本}
```

对于 book 文档类, 默认使用 `twoside` 选项, 但准备文稿时不知道边注将来位于哪一页, 所以应使用上述命令, 其中左边注文本和右边注文本可以是相同内容. 本书的边注实际就是这样输入的.

控制边注样式的参数有

<code>\marginparwidth</code>	边注盒子的宽度
<code>\marginparsep</code>	边柱盒子与正文边界之间的距离
<code>\marginparpush</code>	两个边注盒子之间最小竖直距离

这些参数都是长度, 可用 `\setlength` 命令赋予一个新值.

第九章 数学公式排版的一些技巧

利用第四章的知识, 已经可以比较方便地排版普通的数学公式, 但要排版一些复杂的公式, 还是很麻烦的事情. 本章的前半部分介绍一些更高级的排版技巧, 后半部分介绍宏包 `amsmath`, 它提供了很多命令, 大大增强了复杂公式的排版功能.

§9.1 巧妙使用阵列环境

常见的三角形矩阵

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ & a_{22} & \cdots & a_{2n} \\ & & \ddots & \vdots \\ 0 & & & a_{nn} \end{pmatrix}$$

可以如下输入:

```
\[
\left(\begin{array}{cccc}
a_{11} & a_{12} & \cdots & a_{1n} \\
& a_{22} & \cdots & a_{2n} \\
& & \ddots & \vdots \\
0 & & & a_{nn}
\end{array}\right)
\]
```

下面的矩阵可以看成是分块矩阵, 同时也是加边矩阵:

有的间隔, 代之以花括号内给出的命令, 这里就是 `\hspace{-5pt}`. 目的是为了拉近两列间的距离. 命令 `\[-5pt]` 是用于拉近两行之间的距离.

利用 `array` 环境, 可以排版一些比较复杂的公式, 例如

$$a \neq 0 \left\{ \begin{array}{l} b = 0 \left\{ \begin{array}{ll} c = 0, & y = f(x), \\ c \neq 0, & y = g(x), \end{array} \right. \\ b \neq 0, & y = h(x). \end{array} \right.$$

可以如下输入

```
\[
a\neq\ \bigg\{\begin{array}{l}b=0\ \Big\{
\begin{array}{ll}c=0, & y=f(x),\\
c\neq0, & y=g(x),\end{array}\ \bigg\} b\neq0,\quad y=h(x).
\end{array}
\]
```

`array` 环境不会自动产生公式编号, 若想显示公式编号, 需要使用 `$$...$$` 作行间公式的标记, 并在结束数学模式的标记之前插入下列命令

`\eqno编号` 或 `\leqno编号`

其中 `\eqno` 将编号显示在公式右边, 对齐在页面右边界. `\leqno` 将编号显示在公式左边, 对齐在页面左边界. 例如输入

```
$$
\left(\begin{array}{lcr} lll & ccc & rrr \\ l & c & r \end{array}\right) \eqno(5.3)
$$
```

输出为

$$\left(\begin{array}{lcr} lll & ccc & rrr \\ l & c & r \end{array} \right) \quad (5.3)$$

这是给整个公式一个编号, 并且是人工编号, 若要自动编号, 或每行单独编号, 需使用 `equation` 或 `eqnarray` 环境.

数学公式中出现的多行下标可以看成是一个小型矩阵, 故可使用 `array` 环境排版多行下标. 通常的下标使用角标字体, 对于多行下标最好使用二级角标字体. 因下标命令对 `array` 环境不起作用, 所以需要显式指定字体尺寸, 并相应缩小行距. 这有两种做法, 一是在 `array` 环境中对每个元素指定字体尺寸, 并在换行符后加负的距离; 二是使用文字模式的字体尺寸, 既改变了字的大小, 又自动改变了行距. 第一种方法产生的下标过于靠下, 所以使用第二种方法. 但文字模式的字体尺寸声明不能放在数学模式内部, 因此要用 `\mbox` 进入文字模式, 但 `array` 只能出现在数学模式中, 所以还要用数学模式标记使 `array` 环境成为 `\mbox` 盒子内的行内公式. 下面给出一个例子, 输入

```
\[
  A=\sum_{\mbox{\tiny $\begin{array}{c}
    1\leq i\leq m\\
    1\leq j\leq n\\
    1\leq k\leq p
  \end{array}$}} a_{ijk}
\]
```

输出为

$$A = \sum_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n \\ 1 \leq k \leq p}} a_{ijk}$$

如果我们使用 `amsmath` 宏包的话, 对于多重角标有一个很简单的解决方法, 读者可参看第 164 页.

§9.2 单行公式与多行公式

`eqnarray` 环境总是按三列排版公式, 如果左列的公式很长, 那么右列就很窄了. 一个变通的方法是在左列公式里选一个符号作为中间列, 把它后面的部分作为右列. 不过这样处理时, 原来左列公式的内部会出现较宽的列间距. 最好是有类似于表格环境中的 `\multicolumn` 命令, 把很长的左列单独放在一行上, 而又不影响下面各列的宽度. 遗憾的是在 `eqnarray` 环境中不能使用这个“多列合一”的命令. 好在 `LATEX` 提供了下述命令

```
\lefteqn{公式}
```

它显示作为参数的公式, 但被认为所占的宽度为零, 所在行左边列的列间距仍然存在. 通常在第一行使用这个命令. 例如输入

```
\begin{eqnarray*}
\lefteqn{w+x+y+z = } \\
& & a+b+c+d+e+f+g+ \\
& & h+i+j+k+l+m+n
\end{eqnarray*}
```

输出为

$$\begin{array}{l}
 w + x + y + z = \\
 a + b + c + d + e + f + g + \\
 h + i + j + k + l + m + n
 \end{array}$$

后两行缩进的距离实际就是列间距. 通过在 `\lefteqn{...}` 和 `\\` 之间插入 `\hspace*{长度}` 可以调整缩进深度. 正的 length 增加缩进, 负值则减少缩进.

如果想给多行公式整体一个自动编号并显示在上下居中的位置, 可以使用竖直盒子. 例如公式

$$\begin{array}{ll}
 n!! = 1 \cdot 3 \cdot 5 \cdots n & (n \text{ 为奇数}) \\
 n!! = 2 \cdot 4 \cdot 6 \cdots n & (n \text{ 为偶数})
 \end{array} \tag{9.1}$$

可以如下输入

```
\parbox{80mm}{\begin{eqnarray*}
n!!&=&1\cdot3\cdot5\cdots n\ (\mbox{$n$, 为奇数})\\
n!!&=&2\cdot4\cdot6\cdots n\ (\mbox{$n$, 为偶数})
\end{eqnarray*}} \hfill
\parbox{7mm}{\begin{eqnarray}\end{eqnarray}}
```

两个 `\parbox` 子段盒子左右并列中部对齐, 左边盒子里是一个带*号的(即没有公式编号的)多行公式环境, 用于排版多行公式, 右边盒子里是一个自动编号的多

行公式环境, 用于产生一个公式编号. 不要漏掉两个盒子之间的“弹簧” `\hfill`, 它把公式编号顶到了边上. 如果调用宏包 `amsmath`, 可有更简单的解决方法(见第 156 页).

§9.3 数学模式中的参数

在表格 9.1 中列出了数学模式中的一些参数, 它们的默认值能满足大多数用户的需要, 如果用户有特殊要求, 可以用命令 `\setlength` 重设它们的值. 表中提到的“长”的或“短”的公式不是公式本身的绝对长度, 而是与上方文本行的相对比较. 若上方文本行的结尾到页面左边界的距离大于公式左边界到页面左边界的距离, 则称公式是长公式, 否则称为短公式. 直观看, 被上方最末一行文本遮住一些的公式就是长公式, 一点都未被遮住的就是短公式.

当基本字体尺寸为 10pt 时, 表中参数的预设值分别是 5.0pt, 3.0pt, 8.5pt plus 3.0pt minus 4.0pt, 同左, 0.0pt plus 2.0pt, 4.0pt plus 2.0pt minus 2.0pt, 25.0pt.

表 9.1 数学模式参数

<code>\arraycolsep</code>	array 环境中列间距宽度的一半
<code>\jot</code>	在 <code>eqnarray</code> 和 <code>eqnarray*</code> 环境中增大或减小行间隔
<code>\abovedisplayskip</code>	在长的行间公式与上方文本之间插入的垂直距离
<code>\belowdisplayskip</code>	在长的行间公式与下方文本之间插入的垂直距离
<code>\abovedisplayshortskip</code>	在短的行间公式与上方文本之间插入的垂直距离
<code>\belowdisplayshortskip</code>	在短的行间公式与下方文本之间插入的垂直距离
<code>\mathindent</code>	当选用了文档列选项 <code>fleqn</code> 时行间公式的缩进量

§9.4 数学模式中的字体尺寸

在数学模式中, 有4种控制字体相对大小的字体尺寸. 为行文方便, 在本章中用一个或两个大写字母表示它们:

<code>\displaystyle</code>	D	行间公式的基本尺寸
<code>\textstyle</code>	T	行内公式的基本尺寸
<code>\scriptstyle</code>	S	一级角标的尺寸
<code>\scriptscriptstyle</code>	SS	二级角标的尺寸

对于一个变量如 x , 用 D 尺寸或 T 尺寸显示出来是一样大的.

进入数学模式后, 立即被激活的字体尺寸是 D (行间公式) 或 T (行内公式). 以这两种尺寸为基础, 不同的数学要素使用不同的其它尺寸, 例如当前是 D 尺寸时, 分数使用 T 尺寸, 如果当前是 T 尺寸, 则分数使用 S 尺寸.

四种数学字体尺寸各自对应有一个修正版本, 为方便将修正版本加撇表示. 差别是 D, T, S, SS 的上标要比对应的 D', T', S', SS' 的上标稍稍高了一点点. 在其他情况, 有撇和无撇的字体尺寸是相同的. 不同数学要素选择字体尺寸的规则见表 9.2.

表 9.2 数学字体尺寸规则

激活尺寸	分子	分母	上标	下标
D	T	T'	S	S'
D'	T'	T'	S'	S'
T	S	S'	S	S'
T'	S'	S'	S'	S'
S, SS	SS	SS'	SS	SS'
S', SS'	SS'	SS'	SS'	SS'

有些符号具有两种尺寸, 当前为 D 时, 显示为大符号, 当前为 T 时, 显示为小符号.

在 `array` 环境中, 激活的字体尺寸是 T.

当在分子或上标中显式地指定字体尺寸时, 实际使用的是无撇形式; 而在分母或下标中实际使用的是修正版本.

除了具有两种大小的数学符号, 数学模式内的 4 种字体尺寸的实际大小取决于数学模式外部当前的字体尺寸, 例如当前字体尺寸为 10 pt 时, D 与 T 尺寸也是 10 pt, S 尺寸是 7 pt, SS 尺寸是 5 pt.

§9.5 数学排版的国际标准

国际标准组织(ISO)已经建立了数学排版的一套公认标准. 其中值得注意的有以下几条:

1. 用粗斜体表示向量, 例如: $\boldsymbol{B} \boldsymbol{v} \boldsymbol{\omega}$;
2. 用无衬线斜体表示矩阵与2阶张量, 例如: $\boldsymbol{M} \boldsymbol{D} \boldsymbol{I}$;
3. 特殊常数如 e , i , π , 以及微分算子 d , 都要采用直立体, 以与变量区分;
4. 由数与单位共同构成的度量应看作一个不可分割的整体, 在两者之间有一个小的间隙, 而且单位要用直立体表示, 例如: 5.3 km, 62 kg 等.

其中第4条很容易用命令 `\`, 实现, 例如: 输入 `1.80\,m` 就可得到 1.80 m.

对于第1条, 如果我们已经调入了宏包 `amsbsy` 或 `bm`, 则可用以下的命令之一重新定义 `\vec`:

<code>\renewcommand{\vec}[1]{\boldsymbol{#1}}</code>	(宏包 <code>amsbsy</code>)
<code>\renewcommand{\vec}[1]{\bm{#1}}</code>	(宏包 <code>bm</code>)

否则, 可使用以下命令:

<code>\renewcommand{\vec}[1]{\mbox{\boldmath\$#1\$}}</code>

这样就可输入 `\vec{a}` 得到 \boldsymbol{a} .

对于第3条, 我们可以分别定义3个命令 `\me`, `\mi` 以及 `\dif` 以在数学公式内生成常数 e , i 及微分算子 d :

<code>\newcommand{\me}{\mathrm{e}}</code>
<code>\newcommand{\mi}{\mathrm{i}}</code>
<code>\newcommand{\dif}{\mathrm{d}}</code>

可是直立体的 π 则不是那么容易, 因为 $\text{T}_{\text{E}}\text{X}$ 的字库里没有提供这种字体.

为了达到第2条的要求, 我们必须定义一种新的数学字体, 名为 `\mathsf{sl}`, 再利用这个字体定义一个名为 `\tensor` 的命令 (注意前面一条命令必须出现在导言中):

<code>\DeclareMathAlphabet{\mathsf{sl}}{OT1}{cmss}{m}{sl}</code>
<code>\newcommand{\tensor}[1]{\mathsf{sl}{#1}}</code>

这样我们只要输入 `\tensor A` 就能得到 A .

对于希望能用 ISO 国际标准打印自己的文稿的读者, 不妨把上面的命令做成一个文件, 取名为 `isomath.tex`, 然后在 TeX 源文件的头部用命令 `\input{isomath}` 调入此文件.

§9.6 定理定义的排版

在数学论文或书籍中常常需要用特定的格式展示定理、公理、命题、引理、定义等, 而且带有顺序编号, 为了做到这一点, 需要应用以下命令:

```
\newtheorem{定理环境名}{标题}[主计数器名]
```

例如

```
\newtheorem{theorem}{Theorem}[chapter]
```

定义一个名为 `theorem` 的新的定理环境, 每当这个环境被调用时, 先用黑体字打印 **Theorem**, 再打印一个自动生成的序号, 然后用斜体打印环境内部的文本. 方括号内的可选项应该是章节计数器的名字, 例如这里选的是 `chapter`, 指示把 `chapter` 计数器作为主计数器, 而定理计数器则作为从计数器, 每当节号发生改变时, 就要重置定理计数器的值. 当 `theorem` 环境被定义后, 就可用如下语句进入这个环境:

```
\begin{定理环境名}[附加标题] 文本\end{定理环境名}
```

例如在定义了定理环境 `theorem` 后, 输入以下语句:

```
\begin{theorem}[Fermat] There do not exist integers  $n > 2$ ,  
 $x$ ,  $y$ , and  $z$  such that  $x^n + y^n = z^n$ . \end{theorem}
```

就能得到以下输出:

Theorem 9.1 (Fermat) *There do not exist integers $n > 2$, x , y , and z such that $x^n + y^n = z^n$.*

当标题与附加标题含有中文时, 应添加天元命令 `\黑`, 并用花括号括起来; 对于定理的正文, 则应加上天元命令 `\楷`. 见下例的输入:

```
\newtheorem{dingli}{\黑定理}[chapter]  
\begin{dingli}[\黑费马]{\楷不存在使得  $x^n + y^n = z^n$  的整  
数  $n > 2$ ,  $x$ ,  $y$  以及  $z$ .}  
\end{dingli}
```


其输出为:

定理 9.1 (费马) 不存在使得 $x^n + y^n = z^n$ 的整数 $n > 2$, x , y 以及 z .

`\newtheorem` 的另一种用法是:

```
\newtheorem{定理环境名}[编号]{标题}
```

其中的选项 `编号` 是另一个定理环境的名字, 它与当前的环境共用同一个序号计数器. 例如:

```
\newtheorem{proposition}[theorem]{Proposition}
```

这个命令中间的选项是 `theorem`, 表示 `proposition` 环境与 `theorem` 环境合用一个定理计数器. 这样接在 **Theorem 9.1** 后面的是 **Proposition 9.2**, 而不是 **9.1**.

\LaTeX 有一个加强定理环境功能的宏包 `amsthm`, 只要在导言部分加入命令

```
\usepackage{amsthm}
```

就可以调用这个宏包. `amsthm` 宏包增加了以下功能:

1. 添加了星号命令 `\newtheorem*`, 以创立不带序号的定理环境;
2. 有 3 种预定义的定理格式可供选用:

plain: 标题与编号均为黑体, 定理正文用斜体;

definition: 标题与编号均为黑体, 定理正文用正常字体;

remark: 标题与编号用斜体, 正文用正常字体.

选用格式的命令是 `\theoremstyle{格式}`, 此命令以后的 `\newtheorem` 命令建立的定理环境都采用这个格式, 直至遇到新的 `\theoremstyle` 命令为止.

3. 用户可以使用 `\newtheoremstyle` 命令建立自己的定理环境格式. 其用法可参见 `thmtest.tex` 或 `amsthm.dtx`.
4. 命令 `\swapnumbers` 可使其后创立的定理环境中的序号打印在标题之前, 例如: **1 Theorem**.
5. 对于较短的证明, 可以使用 `proof` 环境, 它先印出一个标题 *Proof*, 然后在结束环境时自动生成证毕记号 \square . 用户可以用命令 `\qedsymbol` 自定义证毕记号, 并使用命令 `\qed` 打印这个记号.

§9.7 amsmath 宏包简介

本章介绍的 `amsmath` 宏包为 2.13 版本, 日期为 2000/07/18. 不同的版本稍有区别, 例如 1.2b 版本没有 `Bmatrix` 环境, `equation` 环境可以排版出多行公式, 而 2.13 版本在这两个环境上正好相反. 当用户不能正常使用本章介绍的命令时, 应检查所用版本是否太旧.

`amsmath` 宏包会自动调入 `amsopn`、`amstext` 和 `amsbsy` 等几个宏包, 如果仅仅需要这几个宏包提供的为数不多的功能, 也可不通过 `amsmath` 而单独调入它们. 为叙述简单, 下面提到 `amsmath` 的功能时一般不再说明是其本身的还是它调入的其他宏包的功能.

`amsmath` 及其调入的宏包提供了许多数学符号和函数名, 还提供如下一些环境用于排版行间公式:

<code>equation</code>	<code>equation*</code>	<code>align</code>	<code>align*</code>
<code>gather</code>	<code>gather*</code>	<code>flalign</code>	<code>flalign*</code>
<code>multline</code>	<code>multline*</code>	<code>alignat</code>	<code>alignat*</code>
<code>split</code>			

除了 `split`, 其他环境都有两种形式, 不带星号的环境具有自动编号的功能, 带有星号时不参与自动编号. 在多行公式的换行命令 `\\` 之前写上 `\notag` 可使该行不带编号. 在两种环境中都可以使用 `\tag{标号}` 人为指定公式的标号, 该标号自动被圆括号括起来, 若使用 `\tag*{标号}`, 则标号不会自动带圆括号. 此处所谓的标号可以是任何字母、数字或其他符号.

除了 `split` 环境之外, 其他环境都进入行间公式模式. `split` 环境不能单独使用, 必须放在其他几个数学环境内部, 但不可用在 `multline` 环境内.

L^AT_EX 的 `eqnarray` 环境仍然可用, 但最好是用 `align` 或 `equation` 与 `split` 代替之, 它们不像 `eqnarray` 在对齐位置产生过大的间隔.

为了调入 (加载) 这个 `amsmath` 宏包, 须在导言区写上语句

`\usepackage[选项]{amsmath}`

通常可不写选项 (即实际使用默认选项). 选项的名称和含义简介如下, 其中具有相反意义的选项放在一起介绍, 默认选项放在前面, 显然这类成对的选项不能同时选用.

`centertags`, `tbtags` 对于 `split` 环境 (见第 156 页). 公式编号默认排在上下居中的位置. 若使用 `tbtags` 选项, 则当公式编号位于左侧时, 编号将被排在公式首行的左侧, 当公式编号位于右侧时, 编号将被排在公式末行的右侧.

sumlimits, nosumlimits 在行间公式中, 求和号的上下限默认排在 Σ 符号的上方和下方. 选项 **nosumlimits** 使上下限排在 Σ 符号右侧. 在具有两种尺寸的符号中, 除了积分号之外, 其他符号均受到该选项的影响.

nointlimits, intlimits 积分号 \int 的上下限默认排在右侧. **intlimits** 选项使上下限排在各种积分符号的上方和下方.

namelimits, nonamelimits 在行间公式中, 对于函数名 $\backslash\det$, $\backslash\gcd$, $\backslash\inf$, $\backslash\lim$, $\backslash\liminf$, $\backslash\limsup$, $\backslash\max$, $\backslash\min$, $\backslash\Pr$ 以及 $\backslash\sup$, 默认极限过程符号排在这些函数名的下方, 形如 $\lim_{n \rightarrow \infty} \sqrt[n]{n} = 1$. 若使用选项 **nonamelimits**, 则极限过程排在上述函数名的右下角, 形如 $\lim_{n \rightarrow \infty} \sqrt[n]{n} = 1$.

reqno, leqno 在行间公式中, 公式编号默认排在公式右侧, 对齐在页面右边界处. 选项 **leqno** 使公式编号排在公式左侧.

fleqn 该选项使所有居中对齐的行间公式左对齐, $\backslash\mathindent$ 的值决定了公式左端与页面左边界之间的距离. 若不使用该选项, 则公式默认是居中对齐的.

上述第 2、3、4 对选项决定了行间公式上下限的标准位置, 可以通过在公式中使用命令 $\backslash\limits$ 或 $\backslash\limits$ 屏蔽掉选项在该公式中的作用, 而使上下限出现在函数名的上下方或右侧. 对于行内公式, 上下限通常排在符号右侧, 同样可以使用上述两个命令人为改变上下限的位置. 例如

$$\backslash\lim\limits_{n \rightarrow \infty} \sqrt[n]{n} = 1$$

总是排成 $\lim_{n \rightarrow \infty} \sqrt[n]{n} = 1$, 而

$$\backslash\lim\limits_{n \rightarrow \infty} \sqrt[n]{n} = 1$$

总是排成 $\lim_{n \rightarrow \infty} \sqrt[n]{n} = 1$.

如果在文档类别的选项中使用了选项 **leqno**、**fleqn**, 则不必在宏包 **amsmath** 的选项中重复.

若只使用宏包 **amsmath**, 会发现 $\backslash\boldsymbol$ 命令对具有两种尺寸的符号不起作用, 此时对这些符号使用 $\backslash\pmb$ 命令, 就可将它们排成黑体. **pmb** 意即 **poor man's bold**, 它并不使用黑体字库, 而是通过微小平移多次重复打印一个符号产生黑体效果, 打印质量略显逊色. 如果仍然调入宏包 **bm**, 则命令 $\backslash\boldsymbol$ 就对所有符号起作用了.

下文中在节标题后附有 (**amsmath**) 时, 介绍的新命令均为加载 **amsmath** 宏包后可以使用的命令. 注意, **TeX** 中的分式型命令 $\backslash\atop$ 和 $\backslash\choose$ 不能再用, 可重新定义 (见第 170 页) 或使用新的命令如 $\backslash\binom$ (见第 169 页).

§9.8 公式中的文本 (amsmath)

在公式中插入文本可以使用命令

```
\text{文本}
```

该命令比 L^AT_EX 原有命令 `\mbox` 功能更强一些, 它会按照所处位置自动改变字体大小. 例如为了排版输出成 $\text{This is}_{\text{subscript}}$, 使用 `\text` 与 `\mbox` 应分别如下输入:

```
This is $_{\text{subscript}}$
```

```
This is $_{\mbox{\scriptsize subscript}}$
```

但要注意, 如果文本是中文, 则不会自动改变大小.

在多行公式的两行之间插入文本可以使用命令

```
\intertext{文本}
```

该命令上方和下方的公式仍处于同一个数学环境之中, 因此可以保持上下的对齐关系不变. 例如输入

```
\begin{align*}
(x+\mi y)(x-\mi y) &= x^2 + \mi xy - \mi xy - \mi^2 y^2 \\
&= x^2 + y^2 \\
\intertext{利用 $\mi^2=-1$, 还可得到}
(x+\mi y)^2 &= x^2 + 2\mi xy - y^2 \\
(x-\mi y)^2 &= x^2 - 2\mi xy - y^2 \\
\end{align*}
```

输出为

$$\begin{aligned} (x + iy)(x - iy) &= x^2 + ixy - ixy - i^2 y^2 \\ &= x^2 + y^2 \end{aligned}$$

利用 $i^2 = -1$, 还可得到

$$\begin{aligned} (x + iy)^2 &= x^2 + 2ixy - y^2 \\ (x - iy)^2 &= x^2 - 2ixy - y^2 \end{aligned}$$

§9.9 单个公式 (amsmath)

单个公式是指整个公式是一个整体, 最多只有一个自动公式编号(使用不带星号的环境), 或没有自动编号(使用带有星号的环境). 单个公式可以只有一行, 也可以有多行. 当单个公式只有一行时, 自动编号时使用 `equation` 环境, 不参与自动编号时使用 `equation*` 环境, 这相当于 L^AT_EX 原有的 `displaymath` 环境.

当单个公式很长, 或不适合排在一行时, 可以把单个公式排成多行, 有如下两种方法.

第一种方法是在 `equation` 环境中使用 `split` 环境, 使每行公式在指定位置对齐, 见下例, 对齐位置用 `&` 指定, 换行符用 `\\`. 输入

```
\begin{equation}
\begin{split}
(a+b)^2 &= a^2 + b^2 + 2ab\\
(a+b+c)^2 &= a^2 + b^2 + c^2 + 2ab + 2ac + 2bc
\end{split}
\end{equation}
```

输出为

$$\begin{aligned}(a+b)^2 &= a^2 + b^2 + 2ab \\ (a+b+c)^2 &= a^2 + b^2 + c^2 + 2ab + 2ac + 2bc\end{aligned}\tag{9.2}$$

如果不写对齐符号, 则输出为右对齐:

$$\begin{aligned}(a+b)^2 &= a^2 + b^2 \\ (a+b+c)^2 &= a^2 + b^2 + c^2 + 2ab + 2ac + 2bc\end{aligned}\tag{9.3}$$

使用 `split` 环境后, 公式编号位于上下居中的位置.

第二种方法是使用 `multline` 环境, 它使第一行靠左, 最末行靠右, 其他行居中对齐. 首末两行与左右边界的距离由 `\multlinegap` 的值决定, 可用 `\setlength` 或 `\addtolength` 改变其值. 公式编号总是位于首行左端或末行右端. 使用命令 `\shoveleft` 或 `\shoveright` 可使中间的公式靠左或靠右对齐. 例如输入

```

\begin{multline}
  \framebox[.5\columnwidth]{第一行(自动靠左)}\\
  \framebox[.4\columnwidth]{自动居中}\\
  \boxed{\sum_{k=1}^{100} k = 5050}\\
  \shoveleft{\framebox[.7\columnwidth]{强制靠左}}\\
  \shoveright{\framebox[.7\columnwidth]{强制靠右}}\\
  \framebox[.5\columnwidth]{第末行(自动靠右)}
\end{multline}

```

输出为

$$\begin{array}{c}
 \boxed{\text{第一行(自动靠左)}} \\
 \boxed{\text{自动居中}} \\
 \boxed{\sum_{k=1}^{100} k = 5050} \\
 \boxed{\text{强制靠左}} \\
 \boxed{\text{强制靠右}} \\
 \boxed{\text{第末行(自动靠右)}}
 \end{array} \tag{9.4}$$

从该例可见, 给公式加框实际是用命令

```
\boxed{公式}
```

如果使用了选项 `fleqn`, 则居中对齐的行都会变为左对齐.

在老的 `amsmath` 版本中, 还可使用 `equation(equation*)` 环境排出多行公式, 每行用 `\cr` (不是 `\\`) 换行, 例如输入

```

\begin{equation}
  (a+b)^2 = a^2 + b^2\cr
  (a+b+c)^2 = a^2 + b^2 + c^2 + 2ab + 2ac + 2bc
\end{equation}

```

输出为

$$\begin{aligned}(a+b)^2 &= a^2 + b^2 \\ (a+b+c)^2 &= a^2 + b^2 + c^2 + 2ab + 2ac + 2bc\end{aligned}\tag{9.5}$$

可见是居中对齐, 并且公式编号总是位于末行. 但若在加载 `amsmath` 时使用了选项 `fleqn`, 则公式是左对齐.

在新的 `amsmath` 版本中, 可以使用 `gather (gather*)` 环境和命令 `\notag` 排出类似效果, 详见下面几节.

§9.10 方程组 (amsmath)

方程组由多个公式组成, 其中每个公式可以占一行也可能占多行. 区分单个公式和方程组, 不能只看行数, 因为单个公式也可以占有多行. 如果整个公式只能有一个自动编号, 则为单个公式, 如果能有多个自动编号, 则为方程组. 两者所用环境不同, 但都有相应不参与自动编号的环境 (带星号的环境).

9.10.1 gather 环境

在 `gather` 环境中每行公式用 `\\` 分隔, 不指定上下对齐的位置. 通常各个公式都是居中对齐, 但若在加载 `amsmath` 时使用了选项 `fleqn`, 则各个公式都是左对齐. 下例中有三个公式, 为了便于看清结构, 公式内容都是文字. 其中一个是 `split` 环境, 该环境是带对齐标记的. 输入

```
\begin{gather}
  \text{这是第一个公式}\\
  \begin{split}
    \text{这是第二个公式,} & \text{使用split环境}\\
    & \text{占据了两行}
  \end{split}\\
  \text{这是第三个公式}
\end{gather}
```

输出为

这是第一个公式 (9.6)

这是第二个公式, 使用split环境 (9.7)

占据了两行

这是第三个公式 (9.8)

9.10.2 align 环境

利用 align 环境可以指定各行公式上下对齐的位置, 通常多在等号或关系运算符处对齐. 例如输入

```
\begin{align}
  圆周率 \pi & \approx 3.1415926535897932384626 \\
  自然对数的底 e & \approx 2.718281828459045 \\
\end{align}
```

输出为

圆周率 $\pi \approx 3.1415926535897932384626$ (9.9)

自然对数的底 $e \approx 2.718281828459045$ (9.10)

align 环境也可使几组公式并排在一起, 此时在同一行上出现分别属于不同组的几个公式, 每个组的公式内应有一个对齐符号 &, 用于该组的上下对齐, 同时在不同组的公式之间也要插入 & 符号, 以分隔这些公式. 同一行上若有 n 个公式, 就必须有 $n+(n-1) = 2n-1$ 个 &, 把一行分成 $2n$ 列. 奇数列总是靠右对齐, 偶数列总是靠左对齐, 于是同一组的公式就靠在一起了. 有时为了叙述方便, 就将这种左右靠在一起的两列称为是一个“列对”. 输入

```
\begin{align*}
  (x^n)' &= nx^{n-1} & (\sin x)' &= \cos x \\
  (a^x)' &= a^x \ln a & (\cos x)' &= -\sin x \\
  && & & (\tan x)' &= \frac{1}{\cos^2 x} \\
\end{align*}
```

输出为

$$\begin{array}{ll} (x^n)' = nx^{n-1} & (\sin x)' = \cos x \\ (a^x)' = a^x \ln a & (\cos x)' = -\sin x \\ & (\tan x)' = \frac{1}{\cos^2 x} \end{array}$$

9.10.3 flalign 环境

`flalign` 环境与 `align` 环境的语法完全一样, 仅仅是输出效果有区别:

`flalign` 环境在同行多个公式之间插入足够的空白以充满一行, 精确地说就是在同一行上对应于偶数个 `&` 的位置自动插入弹性长度, 以充满一行. 例如将上例的环境改为 `flalign*`, 则输出为

$$\begin{array}{ll} (x^n)' = nx^{n-1} & (\sin x)' = \cos x \\ (a^x)' = a^x \ln a & (\cos x)' = -\sin x \\ & (\tan x)' = \frac{1}{\cos^2 x} \end{array}$$

9.10.4 alignat 环境

在 `align` 环境中, 同一行几个列对之间的间距使用默认值 (类似于阵列环境的列间隔), 在 `flalign` 环境中, 这个间距是个弹簧, 总是尽量大, 而在 `alignat` 环境中, 这个间距默认是零, 因此可以通过插入空白长度使列对之间保持指定的间隔. `alignat` 环境需要一个参数, 其值表示同一行中列对的个数, 从前面的介绍可知, 这个值等于同一行中 `&` 的个数加 1 后再除以 2. 例如将上面例子的环境改为 `\begin{alignat*}{2}...\end{alignat*}`, 并且插入间隔, 即如下输入 (其中的 `\label{eq:x}` 是为别处引用所设, 与本环境无关):

```
\begin{alignat}{2}
  (x^n)' &= nx^{n-1} & \hspace{20pt}
  (\sin x)' &= \cos x \\
  (a^x)' &= a^x \ln a & (\cos x)' &= -\sin x \\
  & & & (\tan x)' &= \frac{1}{\cos^2 x} \label{eq:x}
\end{alignat}
```

输出为

$$(x^n)' = nx^{n-1} \quad (\sin x)' = \cos x \quad (9.11)$$

$$(a^x)' = a^x \ln a \quad (\cos x)' = -\sin x \quad (9.12)$$

$$(\tan x)' = \frac{1}{\cos^2 x} \quad (9.13)$$

有时需要在公式后面附加一些上下对齐的文字, 可如下输入:

```
\begin{alignat}{2}
y &= f(x)+g(x) &\quad & \text{(由引理\,2)} \\
&= \sec^2 x && \text{(由\eqref{eq:x}式)}
\end{alignat}
```

输出为

$$y = f(x) + g(x) \quad (\text{由引理 2}) \quad (9.14)$$

$$= \sec^2 x \quad (\text{由(9.13)式}) \quad (9.15)$$

上述两个例子也示例了方程编号的引用方法, 在被引用的方程处写上标记 `\label{标记}`, 在引用处写上 `\eqref{标记}`, 此处标记可以是任何字符串. 将文件编译两次后, 引用处的 `\eqref{标记}` 就会输出为相应方程的编号. 引用命令 `\eqref` 也可改用 `\ref`, 两者的区别是输出时前者自动被圆括号括起来, 而后者并不会自动添加括号. 若引用时用 `\pageref{标记}`, 输出后是被引用标记所在页的页码.

9.10.5 gathered, aligned 和 alignat 环境

前面介绍的方程组环境 `gather`、`align` 和 `alignat`, 无论公式的实际宽度是多小, 它们产生的结构总是占满一行的宽度, 因此不能在它们周围添加括号, 而很多方程组是用花括号连成一组的, 为解决这类问题, 可以使用以 `ed` 结尾的环境 `gathered`、`aligned` 和 `alignat`, 它们的语法和效果虽和不以 `ed` 结尾的环境相同, 但整个结构只占有公式本身实际的宽度, 而不是占满整行, 这样它们就可作为一个“块”结构而放在其它环境之中, 整体作为一个特大的符号与其他符号一同处

理. 这些结构还可加上位置参数, 以决定与其他符号的竖直对齐方式. 需要注意的是这些以 `ed` 结尾的环境本身不再有自动编号的功能, 此外在旧版本中 `aligned` 环境中每行只能有一个对齐符号 `&`, 并且没有 `alignated` 环境. 下面是一个例子:

```
\begin{equation}
\begin{aligned} a&=b+c\\d&=bb+cc \end{aligned}
\Longrightarrow
\begin{gathered}[b] A=aa+bb\\D=c+f \end{gathered}
\Longrightarrow
\begin{aligned}[t] X&=A+aa\\Y&=D+d \end{aligned}
\end{equation}
```

输出为

$$\begin{array}{lcl} & A = aa + bb & \\ a = b + c & \implies & D = c + f \implies X = A + aa \\ d = bb + cc & & Y = D + d \end{array} \quad (9.16)$$

可以在上述块结构旁加各种定界符, 例如输入

```
\begin{equation*}\left.\begin{aligned}
B'&=-\partial\times E \\
E'&=\partial\times B-4\pi j
\end{aligned}\right\} \text{Maxwell 方程}
\end{equation*}
```

输出为

$$\left. \begin{array}{l} B' = -\partial \times E \\ E' = \partial \times B - 4\pi j \end{array} \right\} \text{Maxwell 方程}$$

9.10.6 cases 环境

`cases` 环境专门用于排版左侧带有花括号的方程组, 仅举一例:


```
\begin{equation} f(x)=
\begin{cases} 1 & -1 < x < 1 \\ 0 & \text{其他 } x \end{cases}
\end{cases}
\end{equation}
```

输出为

$$f(x) = \begin{cases} 1 & -1 < x < 1 \\ 0 & \text{其他 } x \end{cases} \quad (9.17)$$

§9.11 矩阵 (amsmath)

`matrix` 环境用于排版矩阵, 本身不带定界符. `pmatrix`、`bmatrix`、`Bmatrix`、`vmatrix` 和 `Vmatrix` 等环境也是用于排版矩阵, 但分别带有定界符 `()`、`[]`、`{}`、`||` 和 `|||`. 默认矩阵最多有 10 列, 各列均居中对齐. 与 L^AT_EX 原有的 `array` 阵列环境 (见第 53 页) 一样, 这些环境必须放在数学模式之中.

可以使用命令

```
\setcounter{MaxMatrixCols}{数}
```

或

```
\addtocounter{MaxMatrixCols}{数}
```

改变最大列数的默认值. 如果想要改变列的对齐方式, 必须使用 `array` 阵列环境.

在低版本 `amsmath` 宏包中没有定义 `Bmatrix` 环境, 用户可自行如下定义:

```
\newenvironment{Bmatrix}%
{\left\lbrace\matrix}\endmatrix\right\rbrace
```

下面例子整体是一个无边的矩阵, 每个元素也都是一种矩阵:

```

\[ \begin{matrix}
\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix} & \\
\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} & \\
\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} & \\
\begin{Bmatrix} 1 & 2 \\ 3 & 4 \end{Bmatrix} & \\
\begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix} & \\
\begin{Vmatrix} 1 & 2 \\ 3 & 4 \end{Vmatrix} & \\
\end{matrix} \]

```

输出为

$$\begin{matrix}
 \begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix} & \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} & \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \\
 \begin{Bmatrix} 1 & 2 \\ 3 & 4 \end{Bmatrix} & \begin{vmatrix} 1 & 2 \\ 3 & 4 \end{vmatrix} & \begin{Vmatrix} 1 & 2 \\ 3 & 4 \end{Vmatrix}
 \end{matrix}$$

从上述例子可见, 可用

```
\\[距离]
```

调整相邻行之间的距离, 对于其他行间公式, 也可如此改变相邻公式的行间隔.

对于出现在行内的矩阵, 一般希望它小一些, 以免撑大行宽, 此时可考虑使用 `smallmatrix` 环境. 例如行内矩阵 $\begin{pmatrix} a & b & c \\ x & y & z \end{pmatrix}$ 是如下输入的:

```

$\bigl( \begin{smallmatrix} a & b & c \\ x & y & z \end{smallmatrix} \bigr)$

```

§9.12 多重数学符号 (amsmath)

9.12.1 多重角标

1 多重角标命令

命令 `\substack` 可以排版多重上标或下标, 两行角标之间用 `\\` 分隔, 排版后角标是居中对齐的. 例如输入

```
\begin{equation}
\sum_{\substack{k_0,k_1,\ldots>0\\
k_0+k_1+\cdots=n}} F(k_i)
\end{equation}
```

输出为

$$\sum_{\substack{k_0,k_1,\dots>0\\k_0+k_1+\dots=n}} F(k_i) \quad (9.18)$$

2 多重角标环境

多重角标环境 `subarray` 不但可以排版多重上标或下标, 而且可以指定角标左对齐 (使用参数 `l`) 或居中对齐 (使用参数 `c`). 下例的角标与上一例相同, 但指定是左对齐. 输入

```
\begin{equation*}
\sum_{\begin{subarray}{l}
k_0,k_1,\ldots>0\\
k_0+k_1+\cdots=n
\end{subarray}} F(k_i)
\end{equation*}
```

输出为

$$\sum_{\substack{k_0,k_1,\dots>0\\k_0+k_1+\dots=n}} F(k_i)$$

9.12.2 多重积分

多重积分符号 `\iint`、`\iiint`、`\iiiiint` 和 `\idotsint` 与普通积分符号 `\int` 一样具有两种尺寸, 上下限既可放在右侧也可放在上下方. 下面是几个例子. 输入

```

\begin{gather*}
\iint\limits_D f(x,y)\,,\,\mathrm{d}x\,\mathrm{d}y\,\mathrm{qquad}
\iiint\limits_V f(x,y,z)\,,\,\mathrm{d}V\,\,\,
\iiint\limits_G f(x,y,z,t)\,,\,\mathrm{d}G\,\mathrm{qquad}
\idotsint\limits_U f(x_1,\,\mathrm{d}ots,x_n)\,,\,\mathrm{d}U
\end{gather*}

```

输出为

$$\begin{array}{cc}
 \iint\limits_D f(x,y) \, \mathrm{d}x \mathrm{d}y & \iiint\limits_V f(x,y,z) \, \mathrm{d}V \\
 \iiint\limits_G f(x,y,z,t) \, \mathrm{d}G & \int \cdots \int\limits_U f(x_1, \dots, x_n) \, \mathrm{d}U
 \end{array}$$

9.12.3 叠置重音符号

数学符号上的重音符号, 例如 \hat{A} 或 \tilde{A} 可以输入为 $\text{\textbackslash hat\{A\}}$ 和 $\text{\textbackslash tilde\{A\}}$. 数学重音符号可以叠置, 例如输入 $\text{\textbackslash hat\{\textbackslash tilde\{A\}\}}$ 可以产生双层重音符号. 当所用 `amsmath` 宏包版本较低时, 叠置的重音符号不会对齐. 为解决这种问题, 宏包中提供了一组用于叠置的重音命令, 与原有的数学重音命令同名, 但第一个字母大写, 它们是

<code>\Hat</code>	<code>\Breve</code>	<code>\Grave</code>	<code>\Bar</code>	<code>\Dot</code>
<code>\Check</code>	<code>\Acute</code>	<code>\Tilde</code>	<code>\Vec</code>	<code>\Ddot</code>

上面的例子若输入成 $\text{\textbackslash Hat\{\textbackslash Tilde\{A\}\}}$, 则输出是 $\hat{\tilde{A}}$. 更多层的叠置也是可以的.

原有的双点重音符号 (`\ddot`) 可以用来表示对时间的二阶导数, `amsmath` 宏包又提供了可以表示三阶和四阶时间导数的重音符号:

`\ddddot` `\dddot`

输入 $\text{\textbackslash dddot\{u\}}$ 和 $\text{\textbackslash dddd\{u\}}$ 输出为 \ddot{u} 和 \dddot{u} .

9.12.4 省略号

1. 三个点的省略号

L^AT_EX 中的省略号命令 `\ldots` 和 `\cdots` 仍然可用, 它们分别产生位于基线上和在一行内上下居中的三个点. 调入 `amsmath` 后, 增加了下列几个省略号命令:

<code>\dots</code>	<code>\dotsb</code>	<code>\dotsc</code>	<code>\dotsm</code>	<code>\dotsi</code>
--------------------	---------------------	---------------------	---------------------	---------------------

当 `\dots` 位于式子中间时, 它会根据尾随其后的符号自动决定省略号的位置: 若后面是关系运算符如等号或二元运算符如 $+$ 、 $-$, 它相当于 `\cdots`, 其他情况相当于 `\ldots`. 见下例:

$\$x_1 + x_2 + \dots + x_n\$$	\Rightarrow	$x_1 + x_2 + \cdots + x_n$
$\$x_1, x_2, \dots, x_n\$$	\Rightarrow	x_1, x_2, \dots, x_n
$\$x_i, \dots + x_j - \dots, x_k\$$	\Rightarrow	$x_i, \cdots + x_j - \cdots, x_k$

但要注意, 当 `\dots` 位于式子末尾时, 没有尾随符号决定它的位置, 它就把省略号排在了基线上, 此时应使用其他几个省略号命令以指定省略号的竖直位置, 命令的最后一个字母表明了命令的含义: `\dotsb` 中的 `b` 表示 `binary` (二元的), 即相当于后面有一个二元运算符, 因此省略号是上下居中的; `\dotsc` 中的 `c` 表示 `comma` (逗号), 不是 `center`, 因此省略号是排在基线上的; `\dotsm` 中的 `m` 表示 `multiplication` (乘法), 因此相当于是 `\dotsb`, 只不过它是用在乘式中, 省略号是上下居中的, 如 $\$A_1 A_2 \dotsm\$$ 生成 $A_1 A_2 \cdots$; `\dotsi` 中的 `i` 表示 `integral` (积分), 省略号对齐在积分号的中部.

2. 长省略号

在矩阵中有时需要跨越几列的省略号, 可使用命令

<code>\hdotsfor{列数}</code>

例如输入

<pre>\[\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \hdotsfor{4} \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix} \]</pre>

可以得到

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

§9.13 分式 (amsmath)

9.13.1 普通分式

L^AT_EX 中的分式命令 `\frac` 仍然可用, 这个分式会随着所处环境而自动改变尺寸. 调入 `amsmath` 后, 增加了两个分式命令

```
\tfrac{分子}{分母}    \dfrac{分子}{分母}
```

它们分别表示在尺寸 `\textstyle` 和 `\displaystyle` 下打印分数. 例如, 无论在行内公式还是在行间公式, `\tfrac{a}{b}` 总是输出成 $\frac{a}{b}$, 而 `\dfrac{a}{b}` 总是输出成 $\frac{a}{b}$.

9.13.2 连分式

调入 `amsmath` 后, 增加了连分式命令

```
\cfrac[位置]{分子}{分母}
```

命令的含义是 `continued fraction`, 其中的可选项位置用于指定分子在分数线上的位置: 省略时是居中, `l` 表示与分数线左对齐, `r` 表示与分数线右对齐. 分母中仍可含有 `\cfrac` 命令, 分子分母自动使用同样大小的字体. 例如输入

```
\[
  a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 +
    \cfrac{1}{a_3 + \cfrac{1}{a_4}}}}
\]
```

输出为

$$a_0 + \frac{1}{a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \frac{1}{a_4}}}}$$

9.13.3 二项式系数

调入 `amsmath` 后, 不能再使用 $\text{T}_{\text{E}}\text{X}$ 中命令 `\atop` 和 `\choose`, 否则编译时会显示出类似下面的出错信息

```
! Package amsmath Error: Foreign command \atop;
  use \frac or \genfrac instead.
```

代替 `\choose` 的是命令:

```
\binom{分子}{分母}
\tbinom{分子}{分母}
\dbinom{分子}{分母}
```

它们都生成二项式系数表达式, 第一个命令会自动选择字体尺寸, 后两个则把当前激活尺寸分别当成是 `\textstyle` 和 `\displaystyle`. 举例如下:

$$\backslash[\backslashbinom{n+1}{k} \backslash] \Rightarrow \binom{n+1}{k}$$

9.13.4 自定义分式类命令

`amsmath` 提供了一个广义分式命令

```
\genfrac{左定界符}{右定界符}{线粗细}{字尺寸}{分子}{分母}
```

其中左定界符和右定界符是放在生成的分式左右两侧的定界符; 线粗细指定分数线的粗细, 空白时表示使用 $\text{L}_{\text{A}}\text{T}_{\text{E}}\text{X}$ 中标准的分数线; 字尺寸指定所用字体的大小, 可为空白或数字 0–3, 空白时表示自动选择字体大小, 数字 0–3 分别表示当前激活字体尺寸为 `\displaystyle`、`\textstyle`、`\scriptstyle` 和 `\scriptscriptstyle`. 前面出现的分数类的命令是如下定义的 (实际定义更复杂一些):

```

\newcommand{\frac}{\genfrac{}{}{}{}{分子}{分母}}
\newcommand{\dfrac}{\genfrac{}{}{}{}0{分子}{分母}}
\newcommand{\tfrac}{\genfrac{}{}{}{}1{分子}{分母}}
\newcommand{\binom}{\genfrac(){}{}{分子}{分母}}
\newcommand{\dbinom}{\genfrac(){}{}0{分子}{分母}}
\newcommand{\tbinom}{\genfrac(){}{}1{分子}{分母}}

```

模仿这些例子, 可以如下重新定义 `\atop` 命令:

```

\renewcommand{\atop}[2]{\genfrac{}{}{0pt}{}{#1}{#2}}

```

注意这与原来的 `\atop` 的 \TeX 风格的用法不同了, 变成了 \LaTeX 风格——分子与分母都是命令后面的参数, 不能放在命令的两边. 公式 (9.18) 现可如下输入

```

\begin{equation}
\sum_{\atop{k_0,k_1,\ldots>0}{k_0+k_1+\cdots=n}} F(k_i)
\end{equation}

```

§9.14 函数 (算子) 名 (amsmath)

在数学公式中, 通常将变量名排成斜体, 将函数名排成正体, 并在函数名的左右带有适当的空白.

9.14.1 已定义的函数名

已定义的函数名 (算子名) 有

<code>\arccos</code>	<code>arccos</code>	<code>\arcsin</code>	<code>arcsin</code>	<code>\arctan</code>	<code>arctan</code>
<code>\arg</code>	<code>arg</code>	<code>\cos</code>	<code>cosh</code>	<code>\cosh</code>	<code>cosh</code>
<code>\cot</code>	<code>cot</code>	<code>\coth</code>	<code>coth</code>	<code>\csc</code>	<code>csc</code>
<code>\deg</code>	<code>deg</code>	<code>\det</code>	<code>det</code>	<code>\dim</code>	<code>dim</code>
<code>\exp</code>	<code>exp</code>	<code>\gcd</code>	<code>gcd</code>	<code>\hom</code>	<code>hom</code>
<code>\inf</code>	<code>inf</code>	<code>\injlim</code>	<code>injlim</code>	<code>\ker</code>	<code>ker</code>
<code>\lg</code>	<code>lg</code>	<code>\lim</code>	<code>lim</code>	<code>\liminf</code>	<code>liminf</code>
<code>\limsup</code>	<code>limsup</code>	<code>\ln</code>	<code>ln</code>	<code>\log</code>	<code>log</code>

<code>\max</code>	<code>max</code>	<code>\min</code>	<code>min</code>	<code>\Pr</code>	<code>Pr</code>
<code>\projlim</code>	<code>projlim</code>	<code>\sec</code>	<code>sec</code>	<code>\sin</code>	<code>sin</code>
<code>\sinh</code>	<code>sinh</code>	<code>\sup</code>	<code>sup</code>	<code>\tan</code>	<code>tan</code>
<code>\tanh</code>	<code>tanh</code>	<code>\varlimsup</code>	$\overline{\lim}$	<code>\varinjlim</code>	\varinjlim
<code>\varliminf</code>	\varliminf	<code>\varprojlim</code>	\varprojlim		

9.14.2 定义新的函数名

为了定义新的函数名, amsmath 提供了命令 (放在导言区)

```
\DeclareMathOperator{\函数名命令}{函数名}
\DeclareMathOperator*{\函数名命令}{函数名}
```

其中第二种 (带星号的) 形式定义的函数名类似于 `\lim`, 会根据所处环境或使用命令 `\limits` 将上下限放置在函数名的上方或下方. 函数名命令与函数名的文字可以不同. 例如在导言区写上命令

```
\DeclareMathOperator{\abc}{abc}
\DeclareMathOperator*{\uvw}{xyz}
```

则有

```
$\abc_1^2 \quad \quad \quad \abc\limits_1^2$       $\Rightarrow$        $abc_1^2$        $abc_1^2$ 
$\uvw_1^2 \quad \quad \quad \uvw\limits_1^2$       $\Rightarrow$        $xyz_1^2$        $xyz_1^2$ 
```

对于某些临时用到的未定义过的函数名, 可直接在公式内部使用命令

```
\operatorname{函数名}      或      \operatorname*{函数名}
```

产生函数名, 星号的含义与前面定义中的星号含义相同. 例如

```
$\operatorname{abc}_a^b$       $\Rightarrow$        $abc_a^b$ 
$\operatorname*{abc}\limits_a^b$       $\Rightarrow$        $abc_a^b$ 
```

§9.15 其他功能 (amsmath)

9.15.1 公式中的空白间隔

在数学公式中产生空白间隔的一般命令是

```
\mspace{数mu}
```

其中单位mu是固定的,不能改用其他单位,也不可省略, $1\text{mu} = 1/18\text{em}$. 看一个例子, 仅为示例\mspace的作用:

$$\text{\textbackslash sum \mspace{-13mu} a \mspace{9mu} b c} \Rightarrow \sum a bc$$

L^AT_EX 提供了几个数学空白命令, amsmath 又增加了几个空白命令, 一同列表如下:

短命令	长命令	空白示例
无空白命令时		$\Rightarrow \neq$
\,	\thinspace	$\Rightarrow \varepsilon$
\:	\medspace	$\Rightarrow \vdash$
\;	\thickspace	$\Rightarrow \vDash$
	\quad	$\Rightarrow \quad \vdash$
	\qqquad	$\Rightarrow \quad \quad \vdash$
\!	\negthinspace	$\Rightarrow \neq$
	\negmedspace	$\Rightarrow \neq$
	\negthickspace	$\Rightarrow \neq$

9.15.2 调整根式指数的位置

amsmath 宏包提供了两个命令

<code>\leftroot{数}</code>	<code>\uproot{数}</code>
---------------------------	-------------------------

用于调整根式指数的位置, 移动的距离单位是一个很小的内部长度, 但不可写上单位. \leftroot 用于指数的左右移动, 正值为向左移动, 负值为向右移动. \uproot 用于指数的上下移动, 正值为向上移动, 负值为向下移动. 示例如下:

$$\text{\textbackslash sqrt[\beta]{f(x)}} \Rightarrow \sqrt[\beta]{f(x)}$$

$$\text{\textbackslash sqrt[\leftroot{-2}\uproot{2}\beta]{f(x)}} \Rightarrow \sqrt[\beta]{f(x)}$$

9.15.3 调整公式编号的竖直位置

当一行公式较长, 使得在同一行上放不下公式编号时, 公式编号将单独占一行被排在公式下方, 此时可在该行公式末(换行符之前)用命令

<code>\raisetag{高度}</code>

调整公式编号的竖直位置, 例如 `\raisetag{6pt}` 将使编号上移 6pt. 若用负的高度, 则是下移位置. 通常是用正的距离, 以减小公式编号行与相应公式行之间的间隔. 观察下例中公式编号与公式之间的距离, 上一个减少了间距, 下一个没有调整距离. 输入

```
\begin{gather}
\pi=4\Bigl(\arctan\frac{1}{2}+\arctan\frac{1}{3}\Bigr)
\doteq 3.141592653589793238462643383279\raisetag{6pt}\\
\pi=16\arctan\frac{1}{5}-4\arctan\frac{1}{239}\doteq
3.141592653589793238462643383279
\end{gather}
```

输出

$$\pi = 4 \left(\arctan \frac{1}{2} + \arctan \frac{1}{3} \right) \doteq 3.141592653589793238462643383279 \quad (9.19)$$

$$\pi = 16 \arctan \frac{1}{5} - 4 \arctan \frac{1}{239} \doteq 3.141592653589793238462643383279 \quad (9.20)$$

9.15.4 特殊的上下标(上下限)

命令

```
\sideset{左侧角标}{右侧角标}主体符号
```

用于在主体符号的两侧放置上下标.

命令

```
\overset{上标}主体符号 \underset{下标}主体符号
```

分别用于在主体符号的上方或下方放置上标或下标. 例如输入

```
\[ \sideset{^a\_tag}{^*_b}\prod\qqquad
\overset{abc}{XY}\qqquad \underset{*}{Z} \]
```

输出为

$$\prod_{\dagger}^a * \quad \overset{abc}{XY} \quad Z_*$$

下列几个命令

<code>\overleftarrow{表达式}</code>	<code>\underleftarrow{表达式}</code>
<code>\overrightarrow{表达式}</code>	<code>\underrightarrow{表达式}</code>
<code>\overleftrightarrow{表达式}</code>	<code>\underleftrightarrow{表达式}</code>

用于在表达式的上方或下方放置水平箭头, 命令的名字已表明了箭头的方向和上下位置, 箭头长短会根据表达式的宽度自动调整. 例如

```
\[ \overleftrightarrow{ABCDE} =
      \underrightarrow{ABC} + \overleftarrow{uvwxyz}
\]
```

输出为

$$\overleftrightarrow{ABCDE} = \underrightarrow{ABC} + \overleftarrow{uvwxyz}$$

当上述箭头出现在角标位置时, 会自动使用较小的尺寸.

命令

```
\xleftarrow[下方表达式]{上方表达式}
\xrightarrow[下方表达式]{上方表达式}
```

在上下居中的位置画出水平箭头, 并把上方表达式放置在箭头的上方. 若有可选下方表达式, 就把它放在箭头的下方. 箭头的长度会自动伸缩以适合表达式的长度. 上下表达式自动使用角标尺寸(含有汉字时需指定汉字大小). 输入

```
\[ A\xleftarrow{1,2\text{\七行交换}} B
      \xrightarrow{II-I\times4}{} C \]
```

输出为

$$A \xleftarrow{1,2\text{行交换}} B \xrightarrow{II-I\times4} C$$

9.15.5 不可断行的区间符

文章中有时用到区间符, 例如语句“调查10-15岁的学生”中的“-”就是区间符, 用于表示一个范围. 为了避免在区间符与随后数字之间可能产生的换行, 应在区间符前加上命令 `\nobreakdash`, 例如输入成 `10\nobreakdash--15`.

§9.16 交换图

我们这里介绍两种画交换图的方法, 一种比较简单, 使用 \LaTeX 的宏包 `amscd`, 可以画出各种方形的交换图. 另一种选择是利用宏包 `diagrams`, 它可以画出相当复杂的交换图. 此外在第115页 §7.7 介绍的绘图软件包 `Xy-pic` 以及 `TikZ&Arrows` 也以画交换图作为设计目标.

9.16.1 用宏包 `amscd` 画交换图

为调入宏包 `amscd`, 必须在导言中加入以下语句:

```
\usepackage{amscd}
```

我们先观察下面的交换图:

$$\begin{array}{ccc} A \amalg B & \xrightarrow{p_A} & A \\ p_B \downarrow & & \uparrow f \\ B & \xleftarrow{g} & Z \end{array}$$

其输入为:

```
\[ \begin{CD}
A\prod B @>p_A>> A \\
@Vp_BVV @AAfA \\
B @<<g<< Z
\end{CD} \]
```

可见 `@>>>` `@<<<` `@AAA` `@VVV` 分别表示向左、向右、向上、向下的箭头, 而出现在第一个 `<` 或 `>` 后面的数学符号会用 `\scriptstyle` 字体打印在水平箭头的上面, 出现在第二个 `<` 或 `>` 后面的数学符号则打印在水平箭头的上面.

类似地, 出现在第一个A或V后面的数学符号会用\scriptstyle字体打印在竖直箭头的左边, 出现在第二个A或V后面的数学符号则打印在竖直箭头的右边.

我们再看下面两个例子:

$$\begin{array}{ccc} G & \xlongequal{\quad} & G' \\ & \parallel & \\ & H & \end{array}$$

其输入为:

```
\[ \begin{CD}
G @= G' \\
@. @| \\
@. H \end{CD} \]
```

这个例子说明命令@=与@|能分别生成水平或竖直的两条平行线. 此外, 如果位于四角的某个对象不出现, 只要输入{}或留空即可, 但是如果某个箭头不出现, 则要输入‘@.’, 不能只是留空.

$$\begin{array}{ccc} G & \xrightarrow{\text{Clifford multiplication}} & H \\ f \downarrow & & \uparrow g \\ G' & \xleftarrow{\beta} & H' \end{array}$$

其输入为:

```
\[ \begin{CD}
G @>\text{Clifford multiplication}>> H \\
@VfVV @AAgA \\
G' @<\phantom{Clifford multiplication}<<\beta< H' \end{CD} \]
```

这个例子让我们看到如何利用TeX的占位命令\phantom使得上下两个箭头有相同的宽度. 这里的\text命令只能在调入amsmath宏包后才能使用, 否则要用\mbox{\scriptsize ...}代替(参见第155页).

9.16.2 用宏包 diagrams 画交换图

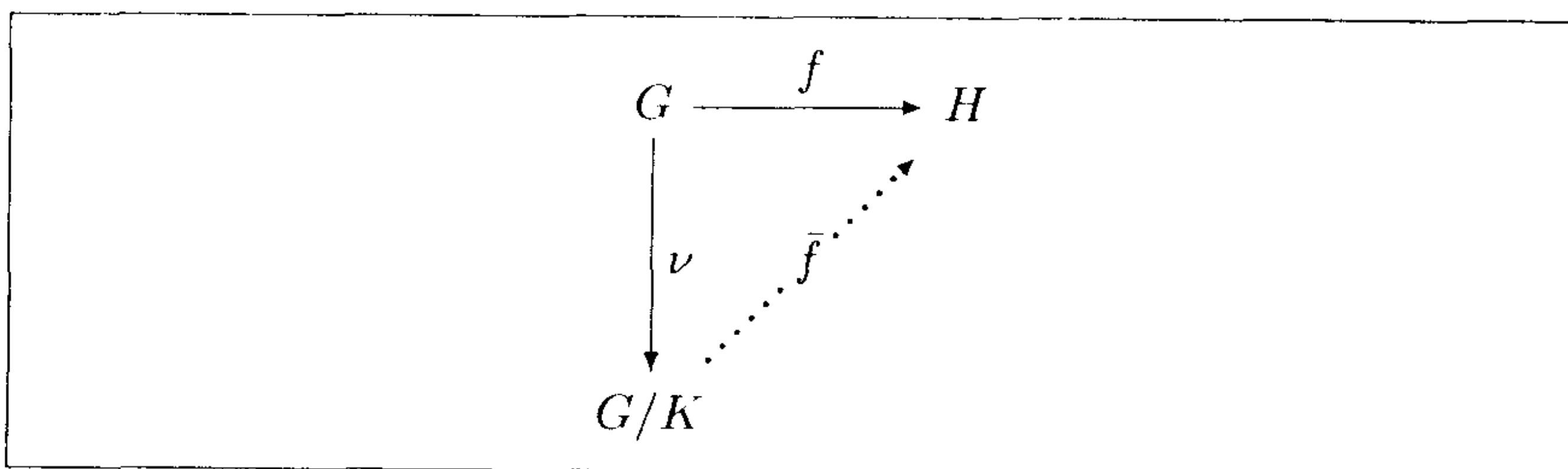
P. Taylor 开发的宏包 `diagrams` 功能十分强大, 要了解它的全部功能, 可以参看文件 `manual.tex`. 这里只介绍一些最容易掌握的命令. 为使用宏包 `diagrams`, 只要先把文件 `diagrams.sty` 复制到当前目录或复制到以下目录(如果没有子目录 `taylor`, 则需要创立):

```
texmf/tex/latex/taylor/diagrams.sty
```

再在导言中加入以下语句:

```
\usepackage{diagrams}
```

现在我们先看一个简单的例子:

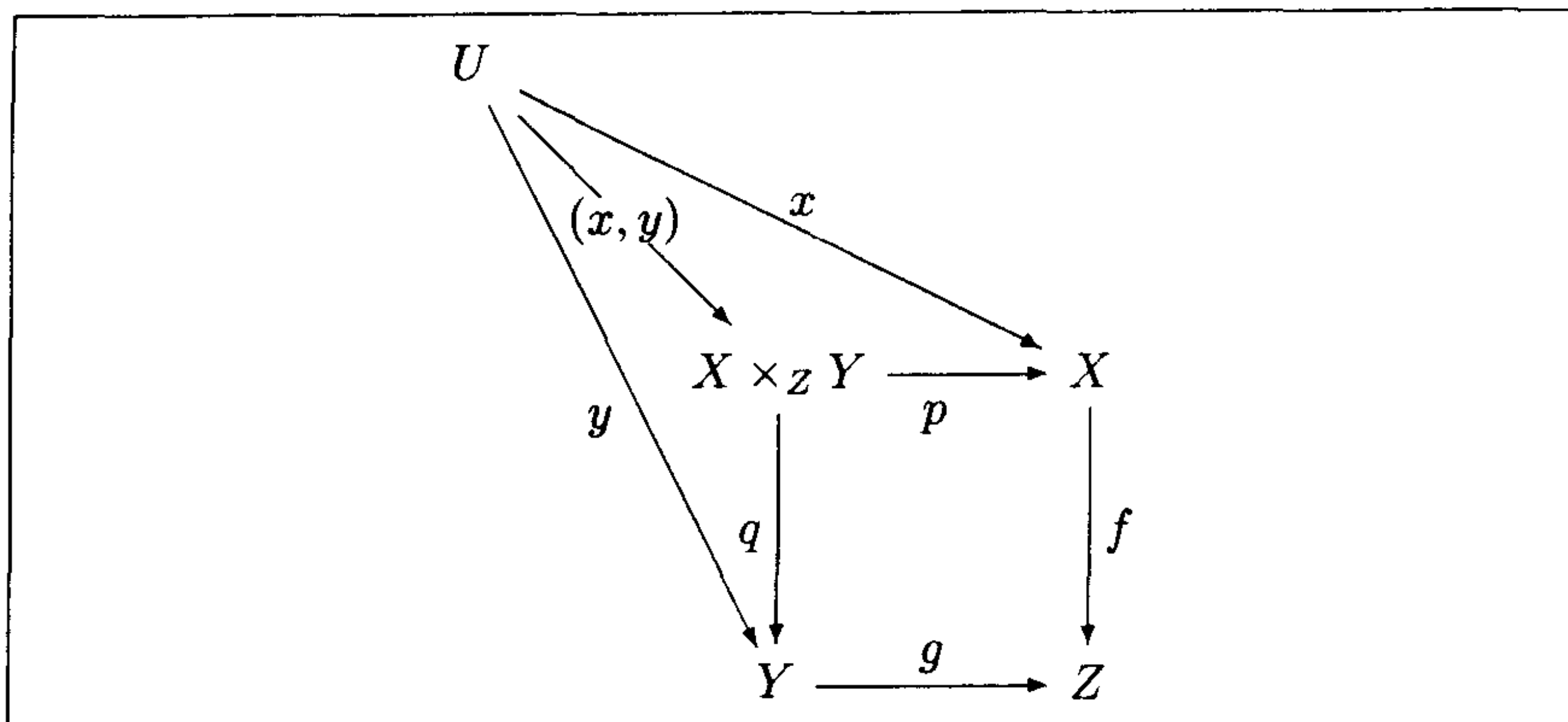


其输入为:

```
\begin{diagram}
G          & \rTo~f          & H \\
\downarrow \nu & \ruDotsto~{\bar f} & \\
G/K        & \end{diagram}
```

从上面的例子可以看出, `diagram` 环境是把一个交换图看成一个表格, 然后再把对象与箭头分别填入相应的格子中. 箭头后面的上标表示注在上面或左面的字符, 下标则表示注在下面或右面的字符, 而注在箭头中间的字符则输入在 `~` 号的后面.

箭头方向的表示方法如下: `\lTo`, `\rTo`, `\uTo`, `\dTo`, `\luTo`, `\ldTo`, `\ruTo`, `\rdTo` 分别表示指向左方、右方、上方、下方、左上方、左下方、右上方、右下方的箭头, 命令 `\rdTo(4,2)` 表示指向左下方的箭头, 目标位于左向第4格、下方第2格. 请读者参看下面的例子:



相应的输入为:

```

\begin{diagram}
U\\
&\rdTo~{(x,y)}\rdTo(4,2)^x\rdTo(2,4)_y\\
&\qquad\qquad\qquad&X\times_Z Y&\rTo_p&X\\
&\qquad\qquad\qquad&\dTo~q&\qquad&\dTo_f\\
&\qquad\qquad\qquad&Y&\rTo~g&Z
\end{diagram}

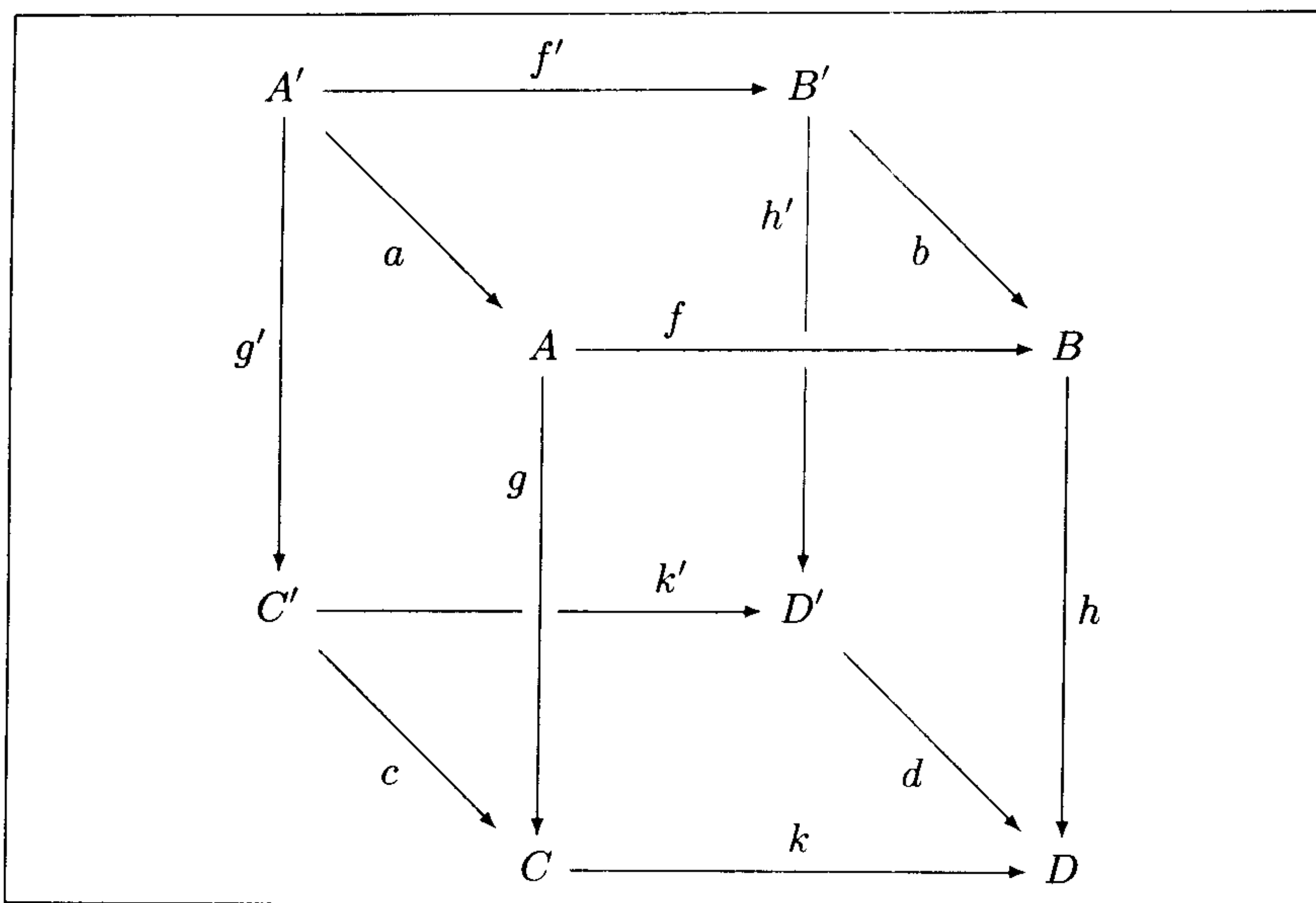
```

下表显示了可以使用的各种箭头式样:

<code>\rTo</code>	\longrightarrow	<code>\rLine</code>	$\rule{1cm}{0.4pt}$
<code>\rEmbed</code>	\hookrightarrow	<code>\rOnto</code>	\twoheadrightarrow
<code>\rDotsto</code>	$\cdots\rightarrow$	<code>\rDashto</code>	\dashrightarrow
<code>\rImplies</code>	\implies	<code>\rMapsto</code>	\mapsto
<code>\rInto</code>	\hookrightarrow	<code>\rProject</code>	\rightarrowtail
<code>\hDots</code>	\cdots	<code>\hDashes</code>	\dashv
<code>\hEq</code>	\equiv		

表中前5种箭头 `\rTo`, `\rLine`, `\rEmbed`, `\rOnto`, `\rDotsto` 可以使用于其他方向, 包括倾斜的方向; 最后3种线条 `\hDots`, `\hDashes`, `\hEq` 只要把命令中的 `h` 改成 `v`, 就能产生竖直方向的线条; 而其余几种箭头 `\rDashto`, `\rImplies`, `\rMapsto`, `\rInto`, `\rProject` 则只能用于水平与竖直的4个方向.

最后再看一个例子:



其输入如下, 注意其中的新命令 `\HonV` 以及 `\VonH` 的意义.

```

\begin{diagram}
  A' & & \rTo^{f'} & & B' & \\
    & \rdTo_a & & & \vLine^{h'} & \rdTo_b \\
\downarrow^{g'} & & A & \rTo^f & \HonV & B \\
    & \downarrow^g & & & \downarrow & \\
  C' & \hLine & \VonH & \rTo^{k'} & D' & \\
    & \rdTo_c & & & & \rdTo_d \\
    & & C & & \rTo^k & D
\end{diagram}
\end{diagram}
  
```

第十章 新字体选择方案 (NFSS)

为了容易地选择和使用各种字体, Frank Mittelbach 和 Rainer Schöpf 于1989年为 \LaTeX 提出了一个新字体选择方案: NFSS (New Font Selection Scheme), 并在1990年初给出了一个初步的测试软件包, 到1993年中, 第二版问世. 1994年4月正式发行的 $\text{\LaTeX} 2_{\epsilon}$ 将NFSS作为重要标准. 在基础篇的§3.2.1和§3.2.2中曾对NFSS作了很简单的介绍.

§10.1 NFSS 中的字体属性

在NFSS中, 字符集按如下五种属性分类: 编码(encoding)、族(family)、系列(series)、形状(shape)和尺寸(size), 选择这些属性的不同组合就选择了不同的字符集. 可以使用下面命令选择这些属性:

<code>\fontencoding{编码}</code>	<code>\fontfamily{族}</code>
<code>\fontseries{权-宽度}</code>	<code>\fontshape{形状}</code>
<code>\fontsize{尺寸}{基线间距}</code>	

编码属性定义了字体中字符的布局. 它的一些可能值列在表10.1中. 通常在一篇文档中不大可能改变编码, 除非要激活其他特殊的字体如西里尔(Cyrillic)字体.

在`\fontfamily`命令中的族参数值表示字体的一组基本属性. 对于CM字体(计算机现代字体 Computer Modern font), 族`cmr`包含所有serif(衬线)字体, 族`cmss`包含所有sans serif(无衬线)字体, 族`cmtt`包含typewriter(打字机)字体, 一些特殊装饰性字体是它们所在族的唯一成员. 在表10.3中根据族和其它属性列出了所有的CM字体.

注意: 字体属性“族”与原来 \TeX 中的同名概念之间没有任何关系. \TeX 中的族是由在数学公式中做为普通文本、角标和二级角标所用的三种不同尺寸的字体组成.

表 10.1 NFSS 的编码

编码	描述	字体样例	页码
OT1	来自于Knuth的原来的文本字体	cmr10	319
OT2	Washington 大学的 Cyrillic 字体	wncyr10	324
T1	Cork(DC/EC)字体	ecrm1000	325
OML	T _E X 数学字母字体	cmmi10	322
OMS	T _E X 数学符号字体	cmsy10	322
OMX	T _E X 数学扩展字体	cmex10	323
U	未知编码	-	

在 `\fontseries` 中的参数值 权-宽度 表示字符的权(笔画的粗细程度)和宽度(字符的宽紧程度)。它们是由表 10.2 中所给的 1 到 4 个字母来定义。表中的“粗”也可称为“黑”。

表 10.2 NFSS 的系列属性

权类			宽度类		
极细	Ultralight	ul	极紧	50%	uc
特细	Extralight	el	特紧	62.5%	ec
细	Light	l	紧 (Condensed)	75%	c
半细	Semilight	sl	半紧	87.5%	sc
中等(正常)	Medium(Normal)	m	中等(正常)	100%	m
半粗	Semibold	sb	半松	112.5%	sx
粗	Bold	b	松 (Expanded)	125%	x
特粗	Extrabold	eb	特松	150%	ex
极粗	Ultrabold	ub	极松	200%	ux

`\fontseries{权-宽度}` 的参数值是由对应于权的字母后接对应于宽度的字母组成。因此 `ebsc` 表示权为特粗, 宽度为半紧, 而 `bx` 意味着权为粗, 宽度为松。同任何非中等(正常)权或宽度组合时, 可以省略字母 `m`; 如果两者都是中等(正常)值, 则只要给出一个 `m` 就行了。

在 `\fontshape` 中, 参数值 形状 是 `n`, `it`, `sl` 或 `sc` 字母组合中的一种, 分别表示正常(直立)、*italic* 斜体、*slanted* 斜体或小体大写字母。

`\fontsize` 属性命令有两个参数值, 第一个参数值 尺寸 表示字体以点(pt)为单位的大小, 但不显式地给出 `pt` 单位, 第二个参数值 基线间距 是上下两条基线

之间的垂直距离. 选择这个属性后, 第二个参数值就成为行距 `\baselineskip` 的新值. 例如, `\fontsize{12}{15}` 选择 12pt 的字体尺寸, 行距为 15pt. 第二个参数值可以给出单位, 例如 15pt, 但如果没有给出单位, 就默认以 pt 为单位.

设置一个或几个属性后, 就可以紧跟着用 `\selectfont` 命令激活具有当前属性的字体. 各种属性是彼此独立的, 改变其中一个, 并不会改变另一个. 例如

```
\fontfamily{cmr} \fontseries{bx} \fontshape{n}
\fontsize{12}{15} \selectfont
```

就选择了一种直立、黑体、松的罗马字体, 尺寸为 12pt, 行距 15pt. 如果后来又用 `\fontfamily{cmss}` 设置一种无衬线字体, 那么在进行下一次 `\selectfont` 调用时, 属性中的权和宽度 `bx`、形状 `n`、尺寸 12(15pt) 继续有效.

等价地, 可以利用下面这条命令来定义除尺寸外的所有属性, 并同时马上激活字体:

```
\usefont{编码}{族}{系列}{形状}
```

由 F. Mittelbach 和 R. Schopf 提供的表格 10.3, 列出了根据族、系列和形状属性对计算机现代字符集的分类. 有相当多的属性组合并不对应某个 CM 字体, 这不是 NFSS 系统的缺陷, 这一设计是为将来考虑的, 它有充分的发展余地. 它也可以应用于正变得越来越普及的 PostScript 字体.

表 10.3 计算机现代字体的属性

系列	形状	外部字体名称示例
计算机现代罗马字体 (<code>\fontfamily{cmr}</code>)		
m	n, it, sl, sc, u	cmr10, cmti10, cmsl10, cmcsc10, cmu10
bx	n, it, sl	cmbx10, cmbxti10, cmbxsl10
b	n	cmb10
计算机现代无衬线字体 (<code>\fontfamily{cmss}</code>)		
m	n, sl	cmss10, cmssi10
bx	n	cmssbx10
sbc	n	cmssdc10
计算机现代打字机字体 (<code>\fontfamily{cmtt}</code>)		
m	n, it, sl, sc	cmtt10, cmitt10, cmsl10, cmtcsc10

形式上是可以任意设置属性的组合的, 然而, 可能不存在一种实际字体对应于所有选择的属性. 如果出现这种情况, 那么当调用 `\selectfont` 时, L^AT_EX 会给

出一条警告信息, 告诉你它用什么字体取代所需字体. 在 `\fontsize` 命令中的字体尺寸属性通常可以取 5、7、8、9、10、10.95、12、14.4、17.28、20.74, 但也可以是其它值. 第二个参数值, 即基线间距, 可以取任何值, 因为它并不是字体固有的性质.

利用 `\begin{document}` 命令, L^AT_EX 给五种属性设置当前特定默认值. 这通常就是标准编码 OT1, 族 `cmr`, 中等(正常)系列 `m`, 正常形状 `n` 和在文档类选项中指定的基本尺寸. 用户可以在导言中改变这些值, 或者用特殊选项把它们设置成不同的值.

§10.2 简化的字体选择命令

属性命令包括 `\fontencoding`、`\fontfamily`、`\fontseries`、`\fontshape` 和 `\fontsize`, 连同命令 `\selectfont`, 都是新字体选择方案中的基本工具. 用户并不需要直接使用这些命令, 而可以利用列在 §3.2 中的命令和声明. 类似于 `\itshape` 这样的字体声明实际定义就是

```
\fontshape{it}\selectfont.
```

族声明(及对应的标准族属性值)为

```
\rmfamily (cmr)   \sffamily (cmss)   \ttfamily (cmtt)
```

这分别对应于计算机现代字体中的罗马、无衬线和打字机字体.

系列声明(及其初始值)为

```
\mdseries (m)     \bfseries (bx)
```

这表明在标准中只提供了正常和粗松两种系列.

形状声明(及相应属性值)为

```
\upshape (n)   \itshape (it)   \slshape (sl)   \scshape (sc)
```

这可以选择直立、斜体、`slanted` 和小体大写字母.

利用 `\normalfont` 命令, 可以随时把族、形状和系列属性值重设为标准值, 但不改变字体尺寸.

对于上面每种字体属性声明, 也存在着一个相应的字体命令, 用以设置其参数文本的字体. 因此 `\textit{text}` 就与 `{\itshape text}` 差不多一样, 区别在于命令中自动包含倾斜校正, 并且命令中的文本参数不能位于两段. 这些命令的清单请参见 §3.2.

注意对编码并没有类似的声明命令, 这是因为通常并不需要在文档中改变编码. 如果要改变编码, 例如要用西里尔(cyrillic)字体时, 需要用到编码 OT2, 可以

进行如下定义:

```
\newcommand{\cyr}{\fontencoding{OT2}\selectfont}
\newcommand{\lat}{\fontencoding{OT1}\selectfont}
```

这样就可以方便地在 OT1 和 OT2 编码之间来回切换了. 其中的命令 `\cyr` 和 `\lat` 可以由用户按个人喜好改用其他字符串.

下面以正文中出现俄文作为改变编码的例子. 有时文章中可能包含一些俄文或其他斯拉夫语言, 这就需要使用西里尔字母, 因此要把当前编码改变成 OT2 编码. 不过在 MiKTeX 的基本安装中并不包括西里尔字母, 如果执行 TeX 出现缺少文件的错误, 就要补充安装 Languages → Cyrillic 模块. 在 OT2 中, 已定义了很多字符编码, 使得直接输入英文字母就会输出相应的西里尔字母, 例如文稿中有俄文单词 алгоритм, 可通过输入 `{\cyr algorifm}` 得到. 下表是一些俄文字母与英文字母的对应表, 因前者个数多于后者, 所以有些俄文字母需输入成英文字母的组合. 表中只有小写的俄文字母, 需要大写字母时只要相应的输入大写英文字母就行了, 若是英文字母组合, 只需第一个字母大写. 从表中看出, 俄文字母基本上是按发音输入相应的英文字母:

а	a	б	b	в	v	г	g	д	d	е	e
ё	\"e	ж	zh	з	z	и	i	й	\U i	к	k
л	l	м	m	н	n	о	o	п	p	р	r
с	s	т	t	у	u	ф	f	х	kh	ц	ts
ч	ch	ш	sh	щ	shch	ъ	\cyrhrdsn			ы	y
ь	\cyrsoftsn	э	\cyrerev			ю	yu	я	ya		

注意输入 `{\cyr ts}` 产生 ц, 如果输入 `{\cyr {t}s}` 则产生 тс. 欲知其中详情, 请参见 MiKTeX 的 `\texmf\doc\latex\cyrillic` 目录内的说明文件.

华盛顿大学设计的西里尔字体名称都以 `wncy` 开头, 后接样式标志 `r` (直立)、`b` (黑体)、`i` (斜体)、`sc` (小体大写) 或者 `ss` (sans serif 字体), 然后是设计尺寸的点数.

与编码有关的一些字体描述文件 (扩展名为 `.fd` 的文件) 将属性组合与具体的字体名称关联在一起, 使得我们也能使用各种不同的西里尔字体. 例如

<code>\textrm{\cyr ABCD abcd 1234}</code>	АБЦД абцд 1234
<code>\textbf{\cyr ABCD abcd 1234}</code>	АБЦД абцд 1234
<code>\textit{\cyr ABCD abcd 1234}</code>	<i>АБЦД абцд 1234</i>
<code>\textsc{\cyr ABCD abcd 1234}</code>	АБЦД АБЦД 1234

<code>\textsf{\cyr ABCD abcd 1234}</code>	АБЦД абcd 1234
<code>{\tiny\cyr ABCD abcd 1234}</code>	АБЦД абcd 1234

§10.3 属性的默认值

在前面一节中的字体属性声明并没有显式地设置属性的值, 而是用某种默认的命令来进行. 例如 `\itshape` 的真正定义是:

```
\fontshape{\itdefault}\selectfont
```

可用的默认命令有:

族: `\rmdefault` `\sfdefault` `\ttdefault`

系列: `\mddefault` `\bfdefault`

形状: `\updefault` `\itdefault` `\sldefault` `\scdefault`

当调用了 `\normalfont` 命令时, 需要定义标准属性值. 它们是由如下四个默认值组成的:

<code>\encodingdefault</code>	<code>\familydefault</code>
<code>\seriesdefault</code>	<code>\shapedefault</code>

默认值应由程序设计者用低级命令去定义它们, 用户只需知道对于 OT1 编码有三个族、两个系列和四种形状的标准属性值可供直接使用.

§10.4 定义新的字体命令

有很多命令可以用来定义新的字体声明和命令, 这些命令主要是为 L^AT_EX 宏包开发者服务的, 但也可以用在普通文档中.

<code>\DeclareFixedFont{\命令}{编码}{族}{系列}{形状}{尺寸}</code>
--

把 `\命令` 定义为一个选择具有指定属性字体的声明. 所有属性都是严格固定的. 这与 `\newfont` 基本等价, 除了这里的字体是由属性而不是由名称确定的.

<code>\DeclareTextFontCommand{\命令}{字体指定}</code>

定义 `\命令` 为一个字体命令, 它按照字体指定设置其参数值. 在内部就是用这条命令来定义所有类似于 `\textbf` 的命令, 而定义 `\textbf` 时字体指定为 `\bfseries`.

<code>\DeclareOldFontCommand{\命令}{文本指定}{数学指定}</code>
--

定义`\命令`为按照`LaTeX 2.09`方式可以用在数学模式中的字体声明. 例如, `\it`的定义为

```
\DeclareOldFontCommand{\it}{\normalfont\itshape}{\mathit}
```

注意此处定义的`\命令`是一个声明, 不是一条命令. 该条语句对于定义与原来版本兼容命令是相当有用的, 但应尽量避免使用.

§10.5 定义新的数学字体命令

10.5.1 数学字母字体

在§4.14.1介绍了可以在数学模式内部调用的数学字体命令有

```
\mathrm \mathit \mathtt \mathsf \mathbf \mathcal
```

这些命令开始字符都是`math`, 其后的字母表明了字体形状. 其中最后一个命令`\mathcal`是数学花体, 它们把作为命令参数的字母设置成特定的字体.

用户可以定义新的数学字体命令, 使字母表中的字母在数学模式中具有新的字体. 命令为

```
\DeclareMathAlphabet{\命令}{编码}{族}{系列}{形状}
```

上面提到的数学字体没有`slanted`斜体, 我们可以定义这种数学斜体, 为此需要定义一个新的数学字体命令(必须放在导言区):

```
\DeclareMathAlphabet{\mathsl}{OT1}{cmss}{m}{sl}
```

这意味着新定义的数学字体命令`\mathsl`选择编码为`OT1`, 族为`cmss`, 权与宽度都是`m`以及形状为`sl`的字体.

在`LaTeX`中已有两种数学变体(version, 或称为种类)的名称, 分别是`normal`和`bold`. 对于已用`\DeclareMathAlphabet`命令定义的命令, 若指定它在某种数学变体下不同的字体属性, 应使用命令(也要放在导言区)

```
\SetMathAlphabet{\命令}{变体}{编码}{族}{系列}{形状}
```

例如若指定新定义的命令`\mathsl`在变体`bold`中的系列属性变为`bx`, 可在导言区放上命令

```
\SetMathAlphabet{\mathsl}{bold}{OT1}{cmss}{bx}{sl}
```


下面看一看新字体命令的输出结果(逗号前是默认的数学字体):

```
$ ABxy, \mathsl{ABxy} $ ABxy, ABxy
\boldmath $ ABxy, \mathsl{ABxy} $ \unboldmath ABxy, ABxy
```

在§9.5中提到国际标准中用无衬线斜体表示矩阵与2阶张量, 但没有这种现成的数学字体命令, 因此定义了一个新的数学字体命令(必须放在导言区):

```
\DeclareMathAlphabet{\mathsfsl}{OT1}{cmss}{m}{sl}
```

若指定它在数学变体bold下不同的字体属性, 可使用命令(也要放在导言区)

```
\SetMathAlphabet{\mathsfsl}{bold}{OT1}{cmss}{bx}{sl}
```

但从表10.3中看到目前还没有一种实际的字体具有上述的属性组合, 因此实际选择的是近似的字体——形状为n. 输出形如 *ABxy*.

10.5.2 数学符号字体

在NFSS下可用下述命令定义一个符号字体名称

```
\DeclareSymbolFont{符号字体名称}{编码}{族}{系列}{形状}
```

它使符号字体名称与指定的属性组合联系起来. 通过这个名称可以定义具体的数学符号.

在标准的L^AT_EX中已定义了如下几个符号字体名称:

```
\DeclareSymbolFont{operators}{OT1}{cmr}{m}{n}
\DeclareSymbolFont{letters}{OML}{cmm}{m}{it}
\DeclareSymbolFont{symbols}{OMS}{cmsy}{m}{n}
\DeclareSymbolFont{largesymbols}{OMX}{cmex}{m}{n}
```

使用已定义的符号字体名称, 可以构造数学字母表和各种不同类型的符号.

```
\DeclareSymbolFontAlphabet{\数学字母表}{符号字体名称}
```

将数学字母表定义成基于符号字体名称的数学字母表. 如在导言区放上命令

```
\DeclareSymbolFontAlphabet{\testi}{letters}
```

那么输入 $\text{\testi{Ax}}$ 就会输出 *Ax*, 这就是符号字体名称 **letters** 对应的属性组合中位于字母表 A、x 位置的字符. 如果在导言区放的命令是


```
\DeclareSymbolFontAlphabet{\testii}{symbols}
```

则输入 $\$ \backslash testii{Ax} \$$ 就会输出 A_x , 它们是符号字体名称 `symbols` 对应的属性组合中位于字母表 A、x 位置 (即第 65 和 120 号位置) 的字符。

定义符号的主要命令是

```
\DeclareMathSymbol{\符号}{类型}{符号字体名称}{位置}
```

此后符号就代表字母表中位于指定位置的字符, 字母表中的字体具有符号字体名称对应的属性组合. 命令中的位置是一个整数, 可以用十进制 (如 65), 也可以用八进制 (如 '81), 或者用十六进制 (如 "41). 类型用于区分符号的类别和功能, 它可以取如下的一个值:

<code>\mathord</code>	普通 (normal) 符号
<code>\mathop</code>	大运算符 (large operator), 如 \sum
<code>\mathbin</code>	二元运算符 (binary operator), 如 \times
<code>\mathrel</code>	关系运算符 (relational operator), 如 \geq
<code>\mathopen</code>	左括号 (opening bracket), 如 $\{$
<code>\mathclose</code>	右括号 (closing bracket), 如 $\}$
<code>\mathpunct</code>	标点符号 (punctuation)
<code>\mathalpha</code>	字母表字符 (alphabet character)

建立数学重音的命令是

```
\DeclareMathAccent{\命令}{类型}{符号字体名称}{位置}
```

其中类型既可以是 `\mathord` 又可以是 `\mathalpha`.

具有两种尺寸的数学符号如定界符 (delimiter)、根号 (radical) 可如下定义:

```
\DeclareMathDelimiter{\命令}{类型}{符号字体名称1}{位置1}
{符号字体名称2}{位置2}
```

```
\DeclareMathRadical{\命令}{符号字体名称1}{位置1}
{符号字体名称2}{位置2}
```

当需要小尺寸符号时, `\命令` 取符号字体名称 1 中位置 1 处的符号, 当需要大尺寸符号时, 则取符号字体名称 2 中位置 2 处的符号。

上述各个命令中没有指定尺寸属性, 这是因为数学模式需要 4 种尺寸 (参见 §4.1 和 §9.3). 这些尺寸是由下述命令指定的 (如要自行指定, 需放在导言区):

`\DeclareMathSizes{正文}{数学文本}{角标}{二级角标}`

其中各个参数都是数字, 表示以点(pt)为单位的字体大小. 如果当前正文尺寸(以点为单位, 下同)恰好等于命令中的参数正文, 则数学模式中的

`\textstyle` `\scriptstyle` `\scriptscriptstyle`

就分别取参数数学文本、角标和二级角标的值. 但若当前的正文文本不等于命令参数中的正文数值, 则数学模式中的尺寸命令就不受任何影响.

一般来讲用户没有必要使用这条命令, 但为了容易理解命令的作用, 现举一例: 设正文标准尺寸是10pt, 在导言区放上命令

`\DeclareMathSizes{12}{20}{14}{10}`

下面是不同输入产生的不同输出:

<code>{ \large \$x^{x^x}=B_{k_j}\$ }</code>	$x^{x^x} = B_{k_j}$
<code>{ \Large \$x^{x^x}=B_{k_j}\$ }</code>	$x^{x^x} = B_{k_j}$
<code>\$x^{x^x}=B_{k_j}\$</code>	$x^{x^x} = B_{k_j}$

这是因为`\large`将字体尺寸设置为12pt, 与命令中的参数匹配. 而`\Large`和正常尺寸都不是12pt, 故不受该命令的影响.

§10.6 在NFSS下指定字体

在NSFF下, 当在文档中指定字体时, 可以先给出所需要的属性组合, 然后调用命令`\selectfont`. 这种字体属性组合与特定的外部字体名称的联系是通过字体定义命令完成的, 这些命令通常存储在扩展名为`def`和`fd`文件中. 下面对其中一些命令作简单介绍. 首先用

`\DeclareFontEncoding{编码}{文本模式命令}{数学模式命令}`

设置一个称为编码的新的编码属性, 以后当选择这种编码的文本字体时, 就会执行文本模式命令, 重定义重音命令或其他依赖于编码的事情. 类似地, 对于这种编码的数学字母就会调用数学模式命令. 也可如下定义默认的文本模式命令和数学模式命令:

`\DeclareFontEncodingDefault{文本模式命令}{数学模式命令}`

这条命令声明了一般的文本和数学模式设置, 更具体的设置出现在前面的命令`\DeclareFontEncoding`中.

对于指定的属性组合, 如果不存在实际的字体与之对应, 下述声明指定了要被取代的属性值:

```
\DeclareFontSubstitution{编码}{族}{系列}{形状}
```

取代是按着形状、系列、族从前到后的次序进行, 编码永远不会被取代. 如果这样取代后仍找不到对应的字体, 那么就用

```
\DeclareErrorFont{编码}{族}{系列}{形状}{尺寸}
```

确定最终应使用的字体.

在特定的编码方案中建立一个新的族使用如下命令:

```
\DeclareFontFamily{编码}{族}{命令序列}
```

每当选择这种编码中该族的字体时, 就会先执行指定的命令序列.

把字体属性与外部字体名称关联起来的主要字体定义声明是

```
\DeclareFontShape{编码}{族}{系列}{形状}{字体定义}{命令序列}
```

每当选择其中的字体时, 就会先执行指定的命令序列.

字体定义由一些尺寸/字体关联项组成, 每一项包含尺寸部分、一个函数、一个可省略参数和一个字体参数. 例如

```
\DeclareFontShape{OT1}{cmr}{m}{n}
{ <5> <6> <7> <8> <9> <10> <12> gen * cmr
  <10.95> cmr10
  <14.4> cmr12
  <17.28> <20.74> <24.88> cmr17}{}
```

说明 `cmr` 族的中等系列、正常形状的成员用外部字体 `cmr5`, ..., `cmr10`, `cmr12` 表示 5-12pt 的尺寸, 以 `cmr10` 放大到 10.95pt 表示 10.95pt 或 11pt 的尺寸, 等等. 如果指定的尺寸不存在, 就用一定限度内最接近的尺寸代替.

尺寸部分放在尖括号内的数表示点数 (pt), 括号内也可以包含范围, 如 `<-10>` 表示所有小于 (但不包括) 10 pt 的尺寸, `<24->` 表示大于或等于 24 pt 的尺寸, `<10-24>` 表示大于或等于 10 pt 并且小于 24 pt 的尺寸, 相当于是数学上的左闭右开区间 $[10, 24)$.

字体定义中可以使用的函数有:

(无) (上面例中的后三行) 加载指名的字体, 放缩到要求的尺寸. 如果在字体名称前面有位于方括号内的选项, 则它是一个附加的放缩因子, 例如

`<11> [.95] cmr10` 加载 `cmr10` 并放缩到 11 pt 的 95%;

gen * 把点数尺寸加到字体参数后, 生成字体名称, 例如

`<5> <12> gen * cmr` 加载 `cmr5, cmr12`;

genb * 把点数尺寸乘以 100 加到字体参数后, 生成字体名称, 用于 DC 和 EC 字体, 例如

`<14.4> genb * ecss` 加载 `ecss1440`;

sub * 替代字体, 这种字体的属性以族/系列/形状的参数形式给出, 例如

`<-> sub * cmtt/m/n` 当没有具有所需属性的字体时, 最好使用这种替代字体; 会显示并在 log 文件中记录一条消息;

subf * 类似于上面的空函数, 但在加载一种替代字体时会给出警告消息;

fixed * 以正常尺寸加载指定的字体, 忽略尺寸部分; 如果给出了可选项参数, 这个参数就是字体要放缩到的尺寸, 例如

`<10> fixed * [11] cmr12` 当要求的是 10 pt 时, 加载 `cmr12` 并放缩到 11 pt.

上面所有函数都可以加一个前缀字母 `s` (表示无声), 以禁止在屏幕上显示信息, 例如 `sub *` 的无声形式是 `ssub *`, 空函数的无声形式是 `s *`.

下面再考虑一个实际的例子: 定义黑、斜打字机字体:

```
\DeclareFontShape{OT1}{cmtt}{bx}{it}{
  <-> ssub * cmtt/m/it }{}
```

因为在计算机现代字体集中没有满足属性组合 `{OT1}{cmtt}{bx}{it}` 的字体, 所以由该命令定义的字体实际将使用替代字体, 包括所有尺寸 (`<->`) 都用系列属性为 `m`、形状属性为 `it` 的打字机字体代替.

字体定义命令既可以放在宏包文件里, 也可以在文档中使用. 但通常是先把每条 `\DeclareFontEncoding` 命令存储在一个文件中, 命名为 编码`enc.def`, 例如 `OT1` 编码对应的文件名是 `ot1enc.def`, 然后把命令 `\DeclareFontFamily` 和 `\DeclareFontShape` 放在扩展名为 `fd` 的文件中, 该文件主名由编码和族名组成, 例如文件 `ot1cmr.fd`. 当选择的编码和族的组合在文档或调用的宏包中没有定义时, `LATEX` 就会自动查找相应的 `fd` 文件并把它调入, 因此无需显式输入这种文件.

但要注意, 如果在当前的格式中不知道编码, 就必须显式地或通过加载 编码`enc.def` 文件调用 `\DeclareFontEncoding`. 加载这种文件的一个方法是调用已有的宏包 `fontenc`, 例如


```
\usepackage[OT2,T1]{fontenc}
```

需要的编码作为选项放在方括号中, 其中最后一个就被置为当前编码.

下面以在NFSS下安装PostScript字体作为一个例子, 一些常用的PostScript字体已经在它们自己的fd文件中作为单独的族定义好了, 例如, ot1ptm.fd就把PostScript的Times字体与一个名为ptm的族关联在一起. 激活这些字体的宏包在文件times.sty中, 其中包括重要的如下几行:

```
\renewcommand{\rmdefault}{ptm}
\renewcommand{\sfdefault}{phv}
\renewcommand{\ttdefault}{pcr}
```

这使得Times ptm成为默认的罗马字体族, 用\rmfamily调用; Helvetica phv成为默认的无衬线字体族, 用\sfddefault调用; Courier pcr成为默认的打字机字体族, 用\ttdefault调用.

下面三行是分别使用命令\textrm、\textsf和\texttt的输出结果, 左边是通常使用的字体, 右边是如上设置的PostScript字体.

ABCDEFabcdef123456	ABCDEFabcdef123456
ABCDEFabcdef123456	ABCDEFabcdef123456
ABCDEFabcdef123456	ABCDEFabcdef123456

§10.7 编码命令

在L^AT_EX中需要使用命令来得到特殊的字符和重音, 例如用\O显示斯堪地纳维亚字符Ø. 这个字符在字体表中的位置与编码有关, 它在OT1中是第31个字符, 而在T1中是第216个字符. 因此当编码改变了时, 需要重新定义所有这样的符号命令. 这可借助于某种编码命令来完成, 这些命令通常与\DeclareFontEncoding命令一同位于编码enc.def文件中.

为了定义一个在不同编码下功能不同的命令, 可以使用

```
\ProvideTextCommand{\命令}{编码}[参数个数][可选项]{定义}
\DeclareTextCommand{\命令}{编码}[参数个数][可选项]{定义}
```

它们的行为同\providecommand和\newcommand命令一样, 只是这样定义的\命令只有当编码被激活时才具有这里给出的定义. 因此对每种编码, \命令具有不同的定义.

```
\DeclareTextSymbol{\命令}{编码}{位置}
```


定义\命令为当编码被激活时在字体表中给定位置处的字符.

```
\DeclareTextAccent{\命令}{编码}{位置}
```

定义\命令为一个重音符号, 当编码被激活时使用字体表中给定位置处的字符作为重音符号.

```
\DeclareTextCompositeCommand{\命令}{编码}{字母}{定义}
```

定义\命令及后接的单个字母的行为是执行定义.

```
\DeclareTextComposite{\命令}{编码}{字母}{位置}
```

定义\命令及后接的单个字母的行为是显示字体表中给定位置处的字符. 这个声明在T1编码中很有用, 因为在T1编码中许多重音字母就是一个单独的符号, 例如第233个字符就是é, 所以在T1编码中可先作声明

```
\DeclareTextComposite{\'}{T1}{e}{233}
```

然后用\'{e}得到é. 在OT1编码中, 这个重音符号是一个单独符号, 位于第19个位置, 因此在OT1编码中应先作声明

```
\DeclareTextAccent{\'}{OT1}{19}
```

然后用\'{e}得到é.

上述的具体声明已包含在文件t1enc.def和ot1enc.def中, 只须激活相应的编码就可直接使用重音命令了.

对于未声明的编码命令可以给出默认定义:

```
\DeclareTextCommandDefault{\命令}[参数个数][可选项]{定义}
\ProvideTextCommandDefault{\命令}[参数个数][可选项]{定义}
\DeclareTextAccentDefault{\命令}{编码}
\DeclareTextSymbolDefault{\命令}{编码}
```

前两条命令给出了适用于所有编码的\命令的默认定义, 后两条则指定\命令把给定的编码作为默认值.

上述一些命令是于1994年12月1日加到L^AT_EX 2_ε中的, 因此在使用这些命令的文件中应包含

```
\NeedsTeXFormat{LaTeX2e}[1994/12/01]
```

§10.8 计算机现代字体

10.8.1 简介

当初 Donald E. Knuth 设计 T_EX 程序时, 同时设计了丰富的字符集或字体, 称它们为计算机现代字体(CM 字体), 后来人们在不断地设计增加新的 CM 字体, 这些字体被分成了几个族. 由于它们是在 NFSS 属性分类系统建立之前出现的, CM 字体命名法并不与 NFSS 完全一致. NFSS 使得用户不必记住 CM 字体名称的细节, 而只需指定属性. 但是使用 NFSS 必须有字体描述文件(扩展名为 `fd` 的文件), 这个文件把属性组合转化为实际的字体名称.

所有 T_EX 字体文件的名称都以 `cm` (表示 “Computer Modern”) 开头, 后接一至四个字母描述字体的样式, 最后是一或两个数字指定设计尺寸的点数, 扩展名是 `tfm`, 意为 “T_EX font metric”.

在 `tfm` 文件中并没有包含生成符号的信息, 而是由字体尺寸信息组成, 这些尺寸包括宽度、高度和深度. 对于 *slanted* 字体, 每个字母还有倾斜校正. 此外, 要为一些字母组合指定与通常不同的间距, 如 `AV` 与 `AV` 的区别, 以及对一些字母进行连写的信息. 还包括字符的斜率(可以为零)、单词间距及其伸展和收缩量、`em` 和 `quad` 间距的大小、句子结尾处的间距. 对于数学和符号字体, 还包括更多的信息.

在安装 L^AT_EX 时已安装了很多 CM 字体, 要想了解究竟有哪些字体, 只需在 T_EX 系统中查找扩展名为 `tfm` 的文件. 下表是部分例子(仅以 10 点为例):

字母	含义	字体名示例
<code>r</code>	直立罗马字体	<code>cmr10</code>
<code>sl</code>	<i>slanted</i> 字体(倾斜了 1/6 的罗马字体)	<code>cmsl10</code>
<code>csc</code>	小体大写直立罗马字体	<code>cmcsc10</code>
<code>ti</code>	文字模式 <i>italic</i> 字体	<code>cmti10</code>
<code>u</code>	‘扶直’了的 <i>italic</i> 字体	<code>cmu10</code>
<code>b</code>	普通黑体	<code>cmb10</code>
<code>bx</code>	扩展黑体	<code>cmbx10</code>
<code>bxsl</code>	黑斜体	<code>cmbxsl10</code>
<code>bxti</code>	黑斜体	<code>cmbxti</code>
<code>ss</code>	直立无衬线字体	<code>cmss10</code>
<code>ssi</code>	倾斜无衬线字体	<code>cmssi10</code>
<code>ssbx</code>	无衬线扩展黑体	<code>cmssbx10</code>
<code>tt</code>	直立打字机字体	<code>cmtt10</code>

<code>tcsc</code>	小体大写打字机字体	<code>cmtcsc10</code>
<code>sltt</code>	倾斜打字机字体	<code>cmslitt10</code>
<code>itt</code>	倾斜打字机字体	<code>cmitt10</code>
<code>mi</code>	数学斜体	<code>cmmi10</code>
<code>mib</code>	数学斜黑体	<code>cmmib10</code>
<code>sy</code>	数学符号	<code>cmsy10</code>
<code>bsy</code>	黑体数学符号	<code>cmbsy10</code>

10.8.2 使用 CM 字体

L^AT_EX 2_ε 提供的现成的字体命令只有有限的几个, 如果用户需要一些特殊的尺寸, 可通过放大或缩小已安装的字体来达到目的. 命令

```
\newfont{\名字}{字体 at 尺寸}
\newfont{\名字}{字体 scaled 因子}
```

定义了一个新的字体名称\名字. 命令中的字体是已有字体的外部文件的主名. 第一种命令把已有字体放缩到指定的尺寸作为自定义字体的大小, 第二种命令中的因子除以 1000 后的值是真正的放缩因子, 把已有字体的尺寸乘以放缩因子得到的值就是自定义字体的大小. 例如需要 123.45pt 的直立罗马字体, 可如下定义

```
\newfont{\myfnt}{cmr10 at 123.45pt}
```

或

```
\newfont{\myfnt}{cmr10 scaled 12345}
```

以后\myfnt就可作为字体尺寸声明使用了, 就象使用\tiny、\huge等声明一样. 但要注意, 这种自定义的字体仅仅具有字体的大小, 而不改变所在段落的行距(基线间距)\baselineskip的值. 如果定义新的字体名称时不带at或scaled参数, 则使用字体原有的尺寸. 通过上述两个命令定义新字体命令, 就可以使用所有已安装的字体了.

在某些T_EX环境中, 可能没有安装OT2编码, 或没有按NFSS格式安装OT2编码, 因此不能象前面介绍的那样使用西里尔字母. 此时可以象上面介绍的那样通过定义新的字体命令来使用西里尔字体, 例如

```
\newfont{\wncy}{wncyr10}
```

定义了一个新的字体声明, 激活 10 pt 的西里尔字体. 使用这个声明后, 输入英文字母就会输出相应的西里尔字母, 例如 алгоритм 可通过输入 {\wncy algorifm} 得到.

每种字体集内的每个字符都有一个序号确定它在该字符集内的位置, 利用序号也可以得到当前字体集内对应的字符. 命令为

`\symbol{序号}`

其中序号可以是十进制, 或是八进制(数字前带前缀“'”), 或十六进制(数字前带前缀“”). 例如, 为了排版输出倒引号“`”, 直接使用键盘上的倒引号不能达到目的, 因为它的输出是英文左单引号“'”, 实际应输入`\symbol{18}`. 又如表示空格“ ”的符号是打字机字体集中的 32 号字符, 可输入成下列三种形式之一:

```
\texttt{\symbol{32}}
```

```
\texttt{\symbol{'40}}
```

```
\texttt{\symbol{"20}}
```

顺便提及, 在 \LaTeX 中使用“`\verb*| |`”命令也可输出空格符号.

第十一章 文档的布局及相互联系

在第五章中介绍了一些常用的文档类别与版式,下面再补充一些与版面布局有关的选项与命令.

§11.1 分栏

在`\documentclass`的可选项中与分栏有关的选项有两个:

<code>onecolumn</code> <code>twocolumn</code>

默认值是`onecolumn`,表示每页只有一栏,即不分栏.选项`twocolumn`使文稿按两栏方式排版,两栏之间的间距以及两栏之间的分隔线的粗细都有默认值,用户也可自行修改.设置栏间距的命令格式是

<code>\setlength{\columnsep}{宽度}</code>

例如命令`\setlength{\columnsep}{3em}`设置栏间距是当前字体中字符M的宽度的3倍.

设置栏间分隔线的命令格式是

<code>\setlength{\columnseprule}{宽度}</code>

例如`\setlength{\columnseprule}{1.5pt}`,设置栏间分隔线的宽度是1.5pt.默认的宽度是零,即不显示分隔线.

表示栏宽的参数是

<code>\columnwidth</code>

对于单栏格式,它与`\textwidth`是相同的.对于两栏格式,其值由`\textwidth`和`\columnsep`所决定.用户不要自行设置栏宽值,但可使用这个值.

上述设置命令如果放在导言区, 其值适用于整个文档, 如果放在正文主体部分, 则只有局部作用, 即其作用终止于下一次的修改, 或者终止于它所在环境或分组的结束.

如果只需要部分页面使用两栏格式, 则不能在文档类别命令中使用两栏选项, 而应在正文中使用命令

```
\twocolumn[通栏文本]
\onecolumn
```

命令 `\twocolumn` 终止当前页, 开始新的一页, 在新页里以双栏格式输出后继文本, 直到遇到 `\onecolumn` 时结束. 如果存在可选项通栏文本, 则在新页的顶部通栏排印通栏文本的内容.

双栏排版时, 总是在排满一页中左边一栏后再排右边一栏, 这意味着当内容较少时, 只能看到一栏而不是两栏, 即不会将内容平均分成两半排成等长的两栏.

不管当前页是否为两栏格式, 命令 `\onecolumn` 总是中止当前页, 开始新的一页, 在新页里以单栏格式输出.

如果想多栏输出, 或者分栏时不另起一页, 可以使用宏包 `multicol`.

§11.2 单、双面

在 `\documentclass` 的可选项中控制单、双面的选项是:

```
oneside      twoside
```

可分别称为单面格式选项和双面格式选项. 选用单面格式 `oneside` 时, 所有页码的打印位置是一致的. 选用双面格式 `twoside` 时, 则把奇数页码打印在页面右边, 把偶数页码打印在页面左边.

单、双面选项并不要求实际打印时必须是单面打印或双面打印, 而仅仅是保证当使用了双面选项而又真正在纸张上双面打印时, 装订成册后页码总是在每页的外侧, 翻阅时容易看到.

对于单面格式, 所有页的左边距都相同, 右边距也相同. 对于双面格式, 则是内侧边距都相同, 外侧边距也都相同.

单面格式是 `article` 文档类和 `report` 文档类的默认选项, 双面格式是 `book` 类的默认选项.

对于双面格式, 可将偶数页称为左页, 奇数页称为右页; 对于单面格式, 偶数页与奇数页的页面样式是相同的, 把它们统统称为右页.

对于 `book` 类, 控制每章开始位置的文档类选项是

`openright openany`

默认值是 `openright`, 使每章都是从奇数页开始, 即翻开书时新的一章总是从右边开始, 这有可能产生空白的偶数页. 选项 `openany` 使每章总是从新的一页开始, 而不管这一页是奇数页还是偶数页.

对于双面格式, 隐含使用了内部命令

`\flushbottom`

使得每页正文底线处在相同的位置上. 对于单面格式, 隐含使用了内部命令

`\raggedbottom`

使得每页正文底线的位置可能稍有上下的变化. 如果在文稿中显式地使用这两个命令, 则可使底线位置与单、双面格式无关.

§11.3 与公式有关的选项

在 `\documentclass` 的可选项中, 常用的与公式有关的选项有三个.

行间公式的编号通常都是显示在右边, 如果想把它们统一改到左边, 可使用选项

`leqno`

但这个选项只对自动编号起作用, 不改变手工编号的位置.

行间公式通常都是居中对齐, 若都改为左对齐, 可使用选项

`fleqn`

当选用了公式左对齐选项 `fleqn` 后, 还可以设置左对齐的位置相对于版心左边界缩进的距离, 命令格式为

`\setlength{\mathindent}{长度}`

与设置栏间距的命令类似, 上述命令放在导言区时对整个文档起作用, 放在正文中时则仅有局部作用.

§11.4 页版式

11.4.1 页面布局

在一个页面上排放文本的区域分成 4 部分: 页眉 (Header)、页主体 (Body)、边注 (Margin Notes) 和页脚 (Footer), 在它们的外部是页边空白区域, 各部分的尺寸以及相互之间的距离参见下一页的图形. 该图形显示的是在 A4 纸上排版 `article` 类文章时的页面布局, 对于 `book` 类或 `report` 类的页面, 除了各部分的尺寸与 `article` 类有所不同外, 边注区的位置还与奇偶页码有关, 使得装订成书后边注总是在外侧. 中文书籍很少有边注.

控制页面版式的命令通常放在导言部分, 命令格式为

```
\pagestyle{版式}
```

其中版式有如下几种:

plain 这是默认的页面版式, 即当导言区没有 `\pagestyle` 时, 相当于在导言区放上了命令 `\pagestyle{plain}`. 排版输出时页面的页眉区是空的, 页码居中放在页脚区.

empty 页面的页眉和页脚都是空的, 并且不显示页码.

headings 页眉由页码和页眉标题组成, 对于 `book` 类和 `report` 类, 页眉标题含有章和节的标题, 对于 `article` 类, 只有节标题. 这些标题是从当前页自动提取的, 但每一章的第一页不显示页眉.

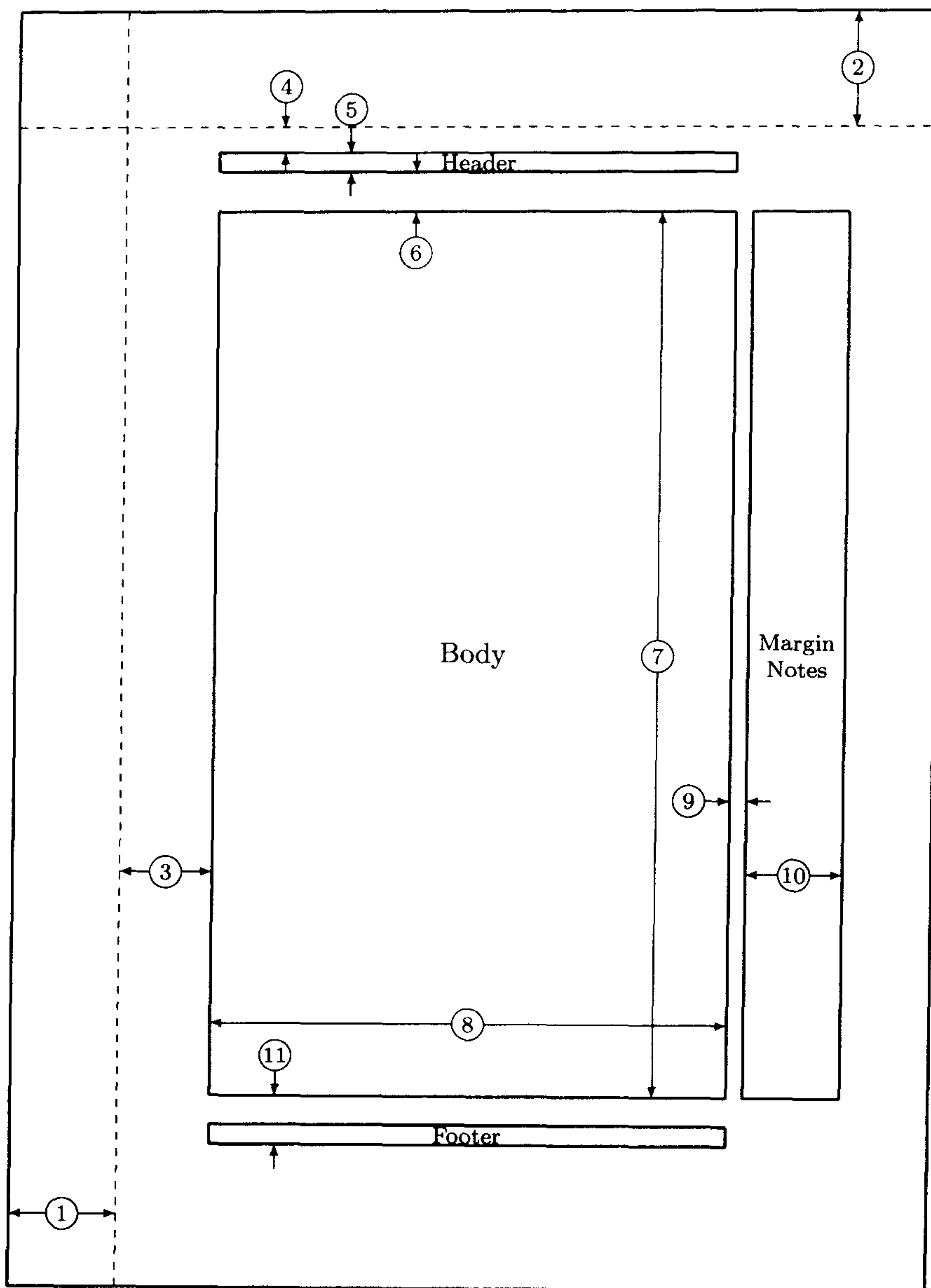
对于不同文档类的双面格式, 左页 (偶数页码) 和右页 (奇数页码) 的页眉内容有所不同, 对于单面格式, 不同文档类的页眉内容也有所不同, 详见下表, 注意单面格式的所有输出页都称为右页.

文档类		左页	右页
book, report	单面格式	---	章
	双面格式	章	节
article	单面格式	---	节
	双面格式	节	小节

对于 `article` 文档类, 如果一页上包含的节 (`\section`) 或小节 (`\subsection`) 多于一个, 则在左页页眉上显示前一个, 在右页页眉上显示后一个.

myheadings 类似于 `headings`, 页眉也是由页码和页眉标题组成, 但标题不是自动提取的, 而是由命令 `\markright` 或 `\markboth` 决定, 见后面的解释.

如果想要使上述版式仅对当前一页起作用, 则应在当前页的文稿中使用如下命令



- | | | | |
|----|-----------------------|----|----------------------------|
| 1 | one inch + \hoffset | 2 | one inch + \voffset |
| 3 | \oddsidemargin = 62pt | 4 | \topmargin = 16pt |
| 5 | \headheight = 12pt | 6 | \headsep = 25pt |
| 7 | \textheight = 550pt | 8 | \textwidth = 345pt |
| 9 | \marginparsep = 11pt | 10 | \marginparwidth = 65pt |
| 11 | \footskip = 30pt | | \marginparpush = 5pt (未显示) |
| | \hoffset = 0pt | | \voffset = 0pt |
| | \paperwidth = 614pt | | \paperheight = 794pt |

`\thispagestyle{版式}`

例如`\thispagestyle{empty}`使当前页的页眉和页脚是空的, 不显示页码. 注意当前页不显示页码并不影响内部计数器, 下一页的页码与用不用这个命令是无关的.

11.4.2 指定页眉内容

对于页面版式`myheadings`, 可用下述命令指定页眉标题内容:

`\markright{右页页眉}`
`\markboth{左页页眉}{右页页眉}`

对于页面版式`headings`, 如果不想显示自动提取的页眉, 也可以使用上述命令指定页眉内容.

对于文档类`article`或使用了文档类选项`oneside`, 所有的输出页都称为右页(不考虑页码的奇偶), 因此应使用`\markright`命令.

对于`book`和`report`文档类, 或使用了文档类选项`twoside`, 则认为偶数页是左页, 奇数页是右页, 这时应使用`\markboth`命令, 但也可使用命令`\markright`重新定义双面格式时的右页页眉.

对于左页, 页码打印在页眉区的左端, 页眉内容对齐在页眉区右端; 对于右页, 页码打印在页眉区的右端, 页眉内容对齐在页眉区左端.

11.4.3 页码

页码的编号可以用阿拉伯数字、罗马数字或拉丁字母, 指定页码样式的命令是

`\pagenumbering{数字形式}`

其中数字形式可取如下值:

arabic	阿拉伯数字(本页页码就是阿拉伯数字)
roman	小写罗马数字(如 i, ii, iii, iv 等)
Roman	大写罗马数字(如 I, II, III, IV 等)
alpha	小写拉丁字母(如 a, b, c, d 等)
Alpha	大写拉丁字母(如 A, B, C, D 等)

默认的数字形式是阿拉伯数字. 需要注意, 每使用一次上述命令, 都会使新页码从1开始. 例如在第一章开始放上命令

```
\pagenumbering{roman}
```

在第二章开始放上命令

```
\pagenumbering{arabic}
```

那么第一章的页码是从i开始的小写罗马数字, 第二章的页码是从1开始的阿拉伯数字.

系统内部有一个页码计数器, 记录当前页的页码. 每生成一页, 计数器就自动加1, 这个新值就是新页的页码. 如果想使起始页码不从1开始, 或者使当前页从新的值开始计数, 可使用如下命令重置计数器的值:

```
\setcounter{page}{页码}
```

其中的页码就是在当前页上要显示的页码.

在book类, 命令

```
\frontmatter
```

把页码切换成罗马数字格式, 并且不再对章进行自动编号, 这个命令通常放在前言和目录前面. 命令

```
\mainmatter
```

把页码切换成阿拉伯数字, 并对章进行自动编号, 这个命令通常放在正文主体前面. 命令

```
\backmatter
```

通常放在参考文献和索引前面, 它不再对章进行自动编号, 但不改变页码数字格式, 不重设页码计数器.

§11.5 目录表及图表清单

在L^AT_EX中, 使用命令

```
\tableofcontents
```

可以自动生成包含章节标题和对应页码的目录表, 并把目录表显示在这个命令出现的地方, 该命令通常是放在文章题目之后, 正文之前.

出现在目录表中的章节层次深度可在导言中用如下命令设定:

```
\setcounter{tocdepth}{数}
```

其中的数与第69页介绍的计数器 `secnumdepth` 的数含义相同, 默认值也相同.

目录表的内容只有在全部文档处理结束后才能完全确定, 因此含有命令 `\tableofcontents` 的文档常常需要编译两到三次. 第一次是在遇到这个命令时, 创建一个与文稿源文件主名同名但扩展名为 `toc` 的目录文件, 然后在后继处理中把遇到的章节信息加到目录文件中, 第二次编译时读入目录文件并打印在对应位置. 如果 `\tableofcontents` 命令未放在文档开头, 那么第二次编译时还要把它前面的章节信息写到目录文件中, 这就需要第三次的编译才能最后完成任务. 当目录文件已经存在, 文档又未修改时, 只编译一次就够了. 不必留心编译了几次, `LaTeX` 系统会提醒用户是否需要再编译一次.

为了把自动编号以外的条目插入到目录表中, 可以使用下列任何一种命令:

```
\addcontentsline{toc}{章节名称}{条目内容}
\addtocontents{toc}{条目内容}
```

第一条命令中的章节名称是去掉前缀倒斜线的章节命令, 用以指定该条目所对应的章节层次. 不同的章节层次在目录表里的默认格式是不同的, 如所用的字体字号是不同的, 缩进的距离也是不同的. 条目内容及其所在页码会按着指定的章节名称的默认格式插入到目录表中.

第二条命令中的条目内容可以含有任何命令和文本, 它们没有任何默认的目录格式, 也不会自动打印该命令所在位置的页码, 当打印目录表时完全是按着条目内容本身包含的命令和文本进行排版.

除了目录表, 插图和表格的列表清单也可由 `LaTeX` 自动生成和打印出来. 清单文件的主名与文稿源文件的主名相同, 插图清单文件的扩展名是 `lof`, 表格清单文件的扩展名是 `lot`, 读入或生成这两种文件的两个命令是

```
\listoffigures    \listoftables
```

列表清单中的条目是根据 `figure` 或 `table` 环境中 `\caption` 命令的参数自动生成的. 也可以象目录表那样手工插入其他条目, 命令的一般格式是:

```
\addcontentsline{清单类型}{格式}{条目内容}
\addtocontents{清单类型}{条目内容}
```

其中的清单类型可以是 `toc` (目录表)、`lof` (插图清单) 或 `lot` (表格清单). 格式可以是章节名称、`figure` 或 `table`.

§11.6 文档的分割处理

对于一篇小文章, 把它放在一个文件里处理起来很方便, 但若是写一本书, 数百上千页的文字, 如果再放在一个文件里, 处理起来就会很慢很笨拙, 此时应该分成几个文件分别处理, 最后再由 `LATEX` 将结果合并起来.

11.6.1 `\input` 命令

使用命令

```
\input{文件名}
```

可以把名为文件名的文件内容读到当前文档的当前位置. 文件名可以带有扩展名, 如果不写扩展名, 则默认扩展名是 `tex`. 但要注意, 如果想读入一个含有中文的文件, 则应先用天元软件进行处理, 把生成的 `tex` 文件读进来, 即在文件名处不要写扩展名, 并且不要更新天元的 `tab` 文件.

`\input` 命令可以放在文档的任何部分, 因此可以把整个导言区放到一个单独的文件中, 当很多文件使用同样的导言区命令时, 不必重复输入, 只要用一条 `\input` 命令把导言区文件读进来就可以了, 这不仅仅是节约工作量, 更重要的是可以保证各个文件导言区命令的一致性.

一个用 `\input` 命令读入的文件中也可以含有 `\input` 命令, 嵌套深度只受计算机能力的限制.

为了得到所有读入文件的清单, 可以把命令

```
\listfiles
```

放在导言区. 当处理完毕, 这个清单会同时出现在屏幕上和记录文件 (`log` 文件) 中.

11.6.2 `\include` 命令

把文档分成几个文件, 对于输入和编辑来说比较方便, 但是当通过 `\input` 命令把它们合并起来后, 处理的仍是整个文档, 这样即使在其中一个文件中进行了一点点改动, 所有的文件都要被重新读入和处理. 因此人们希望能有一种方法, 可以只重新处理被修改的那个文件.

这可以使用命令

```
\include{文件名}
```

它的作用相当于`\clearpage\input{文件名}\clearpage`, 只能用于文档的正文部分. 如果只使用这条命令将多个文件包含进来, 则它并不比`\input`命令好, 反倒有一些局限性, 一是`\include`命令不可以嵌套: 它只能位于主处理文件中. 但`\input`命令可以出现在`\include`进来的文件中. 此外, 它总是开始新的一页然后才读入内容. 不过这并无大碍, 因为人们通常总是在文档应该在开始新页的地方分割文件, 比如章与章之间.

此命令只有与下述命令配合使用才能看出它的优点.

```
\includeonly{文件清单}
```

这条命令只能位于导言区, 文件清单包含了需要被读入的文件. 文件名之间用逗号分开, 省略扩展名`tex`.

在文档中可能包含很多的`\include{文件名}`命令, 只有文件名出现在文件清单中, 或者在导言区没有`\includeonly`命令, 那么`\include{文件名}`才真正读入文件, 等价于`\clearpage\input{文件名}\clearpage`. 如果文件名不包含在文件清单中, 则`\include{文件名}`仅相当于`\clearpage`, 其中内容不被读入. 因此可以通过`\includeonly`命令处理选定的几个文件.

此外这条命令保存有关页面、章节和公式编号等信息. 因此选择部分文件处理时来自于其他文件中交叉引用信息也是可用的, 即`\ref`和`\pageref`命令生成正确的结果. 所有这些值是由前面一次完整处理(用`LATEX`编译文件)确定下来的.

对于长文档, `\include`命令是相当有用的, 它可以节省相当可观的用机时间. 长文档在输入和编辑时通常要分很多步. `\include`命令可以使得在很短时间内有选择地重新处理改动之处.

如果对被选择处理的文件进行了修改, 导致了页码增加或减少, 或者各种编号有了变化, 则其他的文件也需要被重新处理以校正页码和编号, 这只要随后进行一次完整的处理, 或暂时注释掉导言区的`\includeonly`命令就可以做到这一点.

用`\include`读入的文件中不能包含任一`\newcounter`声明. 这并不是一个过分的限制, 因为通常它们都放在导言区.

本书的每一章就是用单独的文件输入的, 它们名称分别是`chap1.ty`, `chap2.ty`, ..., 整体处理文件`all.ty`本身包含如下文本:

```
\input preamble.lt
```



```
\includeonly{.....}
%\typein[\files]{Which files? (for example: chap4, chap5)}
%\includeonly{\files}
\begin{document}
\include{cover}
\frontmatter
\tableofcontents{目\quad录}
\mainmatter
\include{chap1}
\include{chap2}
.....
\backmatter
\printindex
\end{document}
```

其中preamble.lt 存放所有导言区命令. 对于每个ty 文件都先用天元软件处理, 然后再用 L^AT_EX 对 all.tex 进行编译.

11.6.3 人机交互命令

在上一节的例子中有一个命令 \typein, 这是人机交互命令:

`\typein[\命令名]{信息}`

它把信息显示在终端上, 但是它会等待用户从键盘上输入一行文本, 用回车结束. 如果没有可选项 \命令名, 那么这文本就直接插入在处理过程中. 如果包含可选项, 那么就认为它等价于 \typeout{信息}\newcommand{\命令名}{输入的内容}, 这样就会交互式的把输入的内容保存在名称为 \命令名的命令中, 可以在文档的其他部分同其他 L^AT_EX 命令一样调用和执行它. 编译时会在屏幕上显示信息和 @typein=, 等待你的输入.

有的时候希望在处理过程中 L^AT_EX 能在计算机终端上显示出一条消息, 但不需要用户的输入, 这可以用如下命令:

`\typeout{信息}`

这里的信息就代表要显示在计算机屏幕上的文本. 当 L^AT_EX 处理到这条命令时, 就会显示出这段文本. 同时, 消息也写入到 log 文件中. 如果信息中包含命令 (用户

定义的或者 \LaTeX 原有的), 那么命令要被解释, 并把结果显示在屏幕上. 如果命令并不是可显示的, 很可能会造成不良的后果. 要显示命令名而又不执行它, 可在命令的前面加上 \protect 命令.

§11.7 交叉引用

在长的文本中我们经常需要引用在其他地方出现的章节、表格、插图和公式的编号与页码. 对于书籍也需要生成一个索引记录, 它是对整篇文档中特定关键词的引用. 在电子文本处理以前的时代中, 如此的交叉引用和索引意味着要耗费作者或其他工作人员大量的精力和时间. 现在计算机可以承担这一工作的绝大部分. 在以前岁月里, 引用前面文本部分的页码虽然是费事的, 但总是可行的. 但是为了说明将要讲述的内容, 引用还没有写出来的部分就只能局限于章节编号了, 因为页码还不知道, 或者留下空白以备将来填上页码.

编写一本书通常是一个渐进的分阶段的工作. 手稿可能根本就不是按顺序写的, 而且当初稿完成后, 由于作者新的考虑或者来自于评论者的中肯建议, 有可能对它进行很大的修改、修订、删除或插入某些节, 甚至某些章, 交换文本某些部分的顺序更是经常发生的.

\LaTeX 解决了这些改动造成的困难, 它会自动跟踪所有的改动, 保存交叉引用和索引记录的正确信息, 以供在正文的任何地方引用.

11.7.1 交叉引用

在第29页介绍了参考文献的引用方法, 在第56页介绍了公式的引用方法, 文本的其他内容也可交叉引用. 为此, 需要在被引用的位置设置一个标记, 命令为

$\text{\label{标记}}$

其中标记可以是字母、数字和其他符号的任意组合, 但第11页介绍的几个有特殊功能的字符除外.

使用命令

$\text{\pageref{标记}}$

就会把标记所在的页码插入到文本的当前位置.

如果 \label 命令定义在章、节后面或者在公式、枚举罗列(enumerate)等环境中, 或者在 figure 、 table 环境中 caption 的参数值中, 则命令

```
\ref{标记}
```

就会显示出标记所在位置相应对象的编号. 对于由 `\newtheorem` 创建的定理型结构 (第 151 页), 如果 `\label` 命令出现在定理的内容中, 那么定理的编号也可以被引用. 例如在某一页上输入了

```
\begin{theorem} \label{th:fermat} ... \end{theorem}
```

在另外的地方输入文本

```
参见第 \pageref{th:fermat} 页定理~\ref{th:fermat}
```

编译后的输出结果就可能形如“参见第 152 页定理 9”.

标记命令 `\label` 可以放在章节命令的参数值内或紧接在章节命令之后, 相应的引用命令 `\ref` 显示的编号是该章节的编号. 如果标记命令放在正文中, 相应的引用命令显示的是包含标记命令的最内层的那一节的编号.

L^AT_EX 处理文档时, 遇到标记命令 `\label` 就将其参数标记连同相应的计数器的值和当前页码存放在一个辅助文件中, 该辅助文件的主名与正在处理的文档主名相同, 扩展名是 `aux`. `\ref` 和 `\pageref` 命令就是从这个辅助文件得到有关信息的. 第一次编译时仅是收集信息建立辅助文件, 所以不会输出引用信息而是显示两个问号, 再编译一次就成了.

11.7.2 索引记录

为了在文章或书籍的后面打印索引, 简单的做法需要以下一些步骤. 首先, 在输入文本时, 在导言区写上

```
\usepackage{makeidx}
\makeindex
```

其中命令 `\makeindex` 的作用是激活索引功能, 如果用 `%` 把这个命令注解掉, L^AT_EX 将不理睬文稿中有关索引的命令, 似乎它们根本不存在. 这样可在草稿阶段省去调试改错的时间, 等到最后定稿时再把注解号去掉, 激活索引功能.

在需要打印索引的位置, 一般是在文章的最后, 命令 `\end{document}` 之前, 写上

```
\printindex
```

在被索引的条目位置处写上

```
\index{索引条目}
```

这里的索引条目可以是任何文本, 它将来就是出现在后面索引中的一项. 它可以包含字母、数字和各种符号, 也可以是汉字. 对索引条目可使用各种字体. 用 L^AT_EX 编译后, 就会产生一个主名与所处理文件主名相同但扩展名为 `idx` 的文件.

然会使用 `makeindex` 程序处理生成的 `.idx` 的文件, 即把该文件作为程序的命令行参数:

`makeindex 文件名`

可不写扩展名. 运行后产生主名与所处理文件主名相同但扩展名为 `ind` 的文件.

接着再用 L^AT_EX 编译一次主文件就大功告成了, 你会看到类似于本书后每页分成两列打印的索引.

索引页默认的标题是 `Index`, 可以改成中文并设置为大黑字体, 只需使用下述命令

`\newcommand{\indexname}{\大\黑索引}`

接在命令 `\index` 后面的索引条目可以包含字母、数字和各种符号, 甚至包含 T_EX 的命令, 例如: `\index{\section}`, `\index{\[}`, `\index{%}` 都可以, 但是其中出现的花括号必须配对, 例如 `\index{\{\}}` 是允许的, 而 `\index{\{}` 就不可以. 不过 T_EX 的命令在排序的时候被忽略, 而且含有汉字的条目在排序时也会有问题, 这时可使用以下形式的命令:

`\index{排序用条目@打印用条目}`

例如 `\index{put@\verb=\put=}` 表示此条目按 `put` 被排序, 而打印出来的条目却是 `\put`. 这样就能解决汉字的排序问题, 因为 `Makeindex` 的排序原则是参照 ASCII 的编码, 首先是符号, 然后是数字, 最后是字母. 同一个字母, 大写字母排在小写字母之前 (参见本书末尾的索引, 注意我们在排序用条目栏里除去了 ‘\’ 号). 为了使汉字按汉语拼音顺序排在英文字母之后, 我们可用以下的办法输入:

`\index{zzzzsuoyin@索引}`

还要注意字符 ‘!’, ‘@’, ‘|’ 在 `MakeIndex` 里是有特殊功能的命令字符, 因此不能按它们的本来意义出现在索引条目内, 详情可参见 `\texmf\doc\makeindex` 目录内的说明文件.

第十二章 CJK 与 CCT

§12.1 CJK

使 L^AT_EX 能处理中、日、韩文字的 CJK 宏包是 Werner Lemberg 开发的, 目前发布的 4.4.0 版已相当成熟, 因此也是一个很好的选择. 关于 CJK 宏包的安装请参看第 250 页附录一的 A1.3.6 小节, 这里介绍 CJK 的用法. 不过 CJK 尚在不断发完善之中, 有待大家通过使用积累经验. 它的兼容性也还有待检验. 我们这里只介绍与简体汉字有关的内容.

首先我们应该在导言部分加进读入宏包的命令:

```
\usepackage{CJK}
```

这样就定义了两个新的环境:

```
\begin{CJK}[字模编码]{编码}{族}  
.....  
\end{CJK}
```

以及

```
\begin{CJK*}[字模编码]{编码}{族}  
.....  
\end{CJK*}
```

这里的编码与汉字有关的主要有以下几种:

- GB 国标码 GB 2312-1980;
- GBK GBK 大字符集 GB 13000;
- Bg5 繁体汉字大五码;
- Gbt 繁体汉字扩展码 GB/T 12345-1990.

族则取决于你自己的定义, 我们已经替你预先定义了 GBK 编码下的 `song` (宋), `fs` (仿宋), `kai` (楷), `hei` (黑), `li` (隶), `yuan` (圆) 五个族, 这是因为简体中文 Windows

一般提供这 5 种字体. 因此建议你总是使用 GBK 编码. 至于如何增加新的族. 可参看附录一的第 251 页.

可选项字模编码可以忽略.

环境 CJK* 自动忽略汉字间的空格以及换行, 只考虑像 '_ ' 那样受保护的空白, 而 CJK 则与之相反. 根据汉字的习惯, 我们一般应采用 CJK* 环境. 两者间的切换可通过命令

\CJKspace	\CJKnospace
-----------	-------------

来实现, 前者把 CJK* 转换成 CJK, 后者则与之相反.

在 CJK 环境的内部可以再嵌套 CJK 环境, 但是这会耗费 TeX 处理时的堆栈空间, 而且实际上也没有必要, 因为下面的几条命令可以在 CJK 环境的内部切换汉字的编码以及族. 如

\CJKenc{编码}
\CJKfamily{族}

能改变以后出现的汉字的编码及族, 例如 \CJKenc{GB}, \CJKfamily{fs} 等. 为了使用方便, 你可以自己定义新的命令如:

```
\newcommand{\FS}{\CJKfamily{fs}}
```

以后只要发出命令 \FS 就能把字体转换成仿宋体.

在 CJK* 环境里, 遇到西文或数学公式夹在汉字中间时, 可以有两种方式: 或者是紧接着输入, 不留空格, 例如输入 '变量 \$x\$ 的值', 排版结果得到 '变量 x 的值'; 或者是在交界处留空, 例如输入 '变量 _ \$x\$ _ 的值', 排版结果得到 '变量 x 的值'. 前一种情形显得太拥挤, 后一种情形更糟, 前紧后松. 为了解决这个问题, 可使用以下命令:

\CJKtilde

这个命令把不可分割空格 '~' 重新定义为

```
\def~{\hspace{0.25em plus 0.125em minus 0.08em}}
```

这样就能得到合理的间距, 例如输入 '变量 ~\$x\$~ 的值' 可以得到输出 '变量 x 的值'.

如果你想使 '~' 恢复原来的定义, 可以使用以下命令

\standardtilde

如果你偶尔需要使用 '~' 原来的效果, 但又不想使 '~' 恢复原来的定义, 那么可以用以下命令代替原有的 '~':

\nbs

这里的 \nbs 可以看成是 L^AT_EX 命令 \nobreakspace 的缩写.

有了这些命令后, 我们可以看一个例子 (参看 12-1-1.tex):

```
\documentclass{article}
\usepackage{CJK}
\begin{document}
\begin{CJK*}{GBK}{song}
\CJKtilde
\Huge 这是\textbf{一号}\textit{字} (\texttt{\symbol{92}Huge})
\textbf{Fermat~定理:} 对~$n\ge3$,

$$\left\{(x,y,z)\in\mathbf{Q}^3\mid x^n+y^n=z^n, \right. \\ \left. xyz\neq 0\right\}=\emptyset.$$

\huge\CJKfamily{fs} 这是\textbf{二号}\textit{字}
(\texttt{\symbol{92}huge})
.....
\LARGE\CJKfamily{kai} 这是\textbf{三号}\textit{字}
(\texttt{\symbol{92}LARGE})
.....
\Large\CJKfamily{li} 这是\textbf{四号}\textit{字}
(\texttt{\symbol{92}Large})
.....
\large\CJKfamily{yuan} 这是\textbf{小四号}\textit{字}
(\texttt{\symbol{92}large})
.....
\normalsize\CJKfamily{song} 这是\textbf{五号}\textit{字}
(\texttt{\symbol{92}normalsize})
.....
\small\CJKfamily{fs} 这是\textbf{小五号}\textit{字}
(\texttt{\symbol{92}small})
.....
\footnotesize\CJKfamily{kai} 这是\textbf{六号}\textit{字}
(\texttt{\symbol{92}footnotesize})
.....
\scriptsize\CJKfamily{li} 这是\textbf{六号与七号}\textit{字}之间
```

```
(\texttt{\symbol{92}scriptsize})
.....
\tiny\CJKfamily{yuan} 这是\textbf{七号}\textit{字}
(\texttt{\symbol{92}tiny})
.....
\end{CJK*}
\end{document}
```

从例 12-1-1.tex 可以看出西文与汉字的大小可以同步变化。此外, 请注意 `\textbf` 产生汉字黑体的方法是通过对这个汉字重复打印而实现的, 即在水平方向上作轻微移动, 这被称为‘穷人的黑体’ (poor-man’s boldface)。不过 `song` 体对应的黑体则采用 `hei` 体。命令

`\CJKboldshift`

规定了生成黑体的移动距离, 其默认值是 0.015 em, 如果觉得不满意, 可以重新定义, 例如

```
\renewcommand{\CJKboldshift}{0.02em}
```

`\textit` 和 `\textsl` 对汉字的效果是一样的, 都使它变成斜体。其余字体变化命令均对汉字没有影响。光盘上还给出了原来使用天元的例子的 `cjk` 版, 它们的文件名是 `?-?-?cjk.tex`, 表示对应于 `?-?-?.ty`。读者可对这些文件一一编译, 以观察其效果。

下面再介绍一些有用的 CJK 命令。

`\CJKglue`

这个命令的原始定义是:

```
\newcommand{\CJKglue}{\hskip 0pt plus 0.08\baselineskip}
```

它于必要时在汉字之间插入一个附加的空隙, 以解决行的超长。如果你遇到行超长无法解决时, 可以修改这个命令, 增大它的值, 以加强其调节功能。

`\CJKkern`

此命令阻止在这两个汉字间换行, 它取消了命令前面的 `\CJKglue` 的作用。

此外, `verbatim` 环境在 CJK 内仍然有效。

更进一步的用法请参看 `texmf\doc\latex\Cjk` 目录中的 `commands.doc` 以及 `CJK.doc`, 这是两个文本文件, 不需使用 Word。

§12.2 CCT 系统简介

CCT 系统是由中国科学院计算数学与科学工程计算研究所张林波等人开发研制的 T_EX 中文接口, 可从

`ftp://ftp.cc.ac.cn/pub/cct`

免费下载 CCT 系统软件.

本文仅对 CCT 系统作简略的介绍, 读者欲了解详细使用说明, 请参阅该软件自带的参考手册.

12.2.1 CCT 系统概况

CCT MSDOS 版可在各种具有 512K 以上内存并配置有硬盘的 PC 机上运行. CCT 系统支持下列图形方式: CGA (640×200), EGA (640×350), VGA (640×480), CGE (Color 400, 640×400) 及一些增强 VGA 方式及 VESA 方式 (800×600, 1024×768, 等等).

CCT 支持的输出设备有: 各种 24 针打印机; HP, 北佳 (PECAN) 及与之兼容的激光打印机; HP, CANON 喷墨打印机; 长春 JZJ、IPX 激光照排机等.

CCT MSDOS 版运行所需要的环境为: MSDOS 3.0 以上版本, 任何基于 GB2312 编码的中文编辑系统 (用于汉字的输入和编辑), 以及 2.0 以上版本的 T_EX 系统.

用 CCT 系统进行排版的过程主要包括以下几步:

1. 准备 CCT 源文件, 建议用 `ctx` 作文件扩展名. 源文件中包含排版内容及排版命令. 作为例子, 设文件名为 `test.ctx`.
2. 用 CCT 的预处理程序 `CCT.EXE` 将 CCT 源文件转换为 T_EX 源文件:

```
C>cct test
```

3. 用 T_EX 软件 (以 L^AT_EX 为例) 对得到的 T_EX 源文件进行排版处理:

```
C>latex test
```

4. 用 CCT 的屏幕显示驱动程序显示排版结果:

```
C>dviscr test
```

5. 用 CCT 的打印驱动程序 (以 HP LaserJet+ 激光打印机为例) 打印排版结果:

```
C>dviljp test
```

12.2.2 CCT 的排版命令

CCT 源文件与 T_EX 源文件的格式一样. 正文(包括中文、英文等等)可以直接输入而指令仍然由英文字母构成. 除了标准的 T_EX 命令外, CCT 系统定义了下面一些与汉字有关的命令:

```

\ziti      \zihao      \ziju
\songti    \heiti      \kaishu    \fangsong  \biaosong
\ccwd      \ccht       \ccdp
\pushziti  \popziti

```

由于这些指令定义在文件 CCHEAD.STY 中, 因此用户在 CCT 源文件开头需要加入下面的指令:

```
\input cthead.sty
```

或者 (L^AT_EX 2_ε)

```
\usepackage{cchead}
```

来调入这些命令. 下面分别说明这些命令的含义及用法.

\zihao 命令用来选择汉字的大小(即字号), 后面必须用一个括在大括号中的整数给出选用的字号. 允许用户使用十种不同大小的字号, 它们是:

CCT 指令	对应的字号	CCT 指令	对应的字号
\zihao{0}	初号(特号)	\zihao{-4}	小四号
\zihao{1}	一号	\zihao{5}	五号
\zihao{2}	二号	\zihao{-5}	小五号
\zihao{3}	三号	\zihao{6}	六号
\zihao{4}	四号	\zihao{7}	七号

CCT 系统在字号定义文件 CCT.DAT 中定义了各种字号的大小以及相应的字距与行距, 如果修改了这个文件, 则需重新运行一次 CCTINIT.EXE 程序来生成相应的 TFM 文件及 CCHEAD.STY 文件.

\ziti 命令用来设置汉字的字体, 后面跟随一个括在花括号中的字母(不区分大小写). 通常 \ziti{A}, \ziti{B}, \ziti{C}, \ziti{D} 和 \ziti{E} 分别保留为宋体、黑体、楷书体、仿宋体和标宋, 它们可分别简写为 \songti, \kaishu, \heiti, \fangsong 及 \biaosong. 如果用户想要使用这五种标准字体之外的字体, 则可用命令 \ziti{F}, \ziti{G} 等等来选择, 并在文件 CCFONTS.DEF 中给出相应的字库文件名. 用户最多可同时使用 26 种不同字体.

`\ccht`、`\ccdp`、`\ccwd` 命令 分别给出当前字号的字高 (height)、字深 (depth) 和字宽+字距 (width+inter word space). 它们可做为长度单位使用.

`\ziju` 命令用来改变缺省的汉字字距. 它包含一个参数给出字距与字宽之比. 例如, `\ziju{0.25}` 将字距设为字宽的 0.25 倍 (通常使用的正常字距可使用默认值或取为 0.06).

`\pushziti`, `\popziti` 命令均没有参数. `\pushziti` 命令在输出时使得驱动程序将当前字体压入一个栈中, 而 `\popziti` 命令使得驱动程序将当前字体恢复成上一次 `\pushziti` 命令所保存的字体 (栈的最大深度为 16). 这两条命令通常放在浮动体内容的两端, 以保证浮动体内的字体变化不会影响浮动体之外的文本.

CCT 提供了一个拼字程序 `PZ.EXE` 用来拼造区位表中没有的字, 用户可利用它来建造用户字库. 具体使用方法请参阅 CCT 系统附带的使用手册.

在用户排版源文件中, 中西文之间留不留空格都不影响排版输出的结果, 例如 CCT 的源文件中含有下面的内容: “中文English汉字” 或 “中文_English_汉字”, CCT 的预处理程序将它们都是转换成:

```
{\CC .....} English {\CC .....}
```

其中“...”代表汉字转换后产生的字符. 这样的处理使得在汉字与西文字符间总是留有相当于一个西文空格的空. 但有时用户不希望有多余的空格, 目前 CCT 的处理方式尚无法自动避免这个问题. 作为一个解决方案, CCT 作者建议用户定义下面两条命令:

```
\def\CS{{\CC\ }}
```

```
\def\ccnospace#1{\leavevmode\unskip #1\ignorespaces }
```

其中命令 “\CS” 留出相当于汉字之间字间距的空, 而 “\ccnospace” 则用来取消多余的空.

中文字号控制命令 (`\zihao`) 只改变中文的大小, 西文字号控制命令只改变西文大小. 为了使 \LaTeX 的字号控制命令如 `\large` 等能同时控制中文和西文的大小, CCT 修改了 \LaTeX 的一些系统文件, 提供了两个适合中文排版的文档类 `cctart` 和 `cctbook`, 前者适用于文章的排版, 后者则适用于书籍的排版. 排版中文文章或书籍时, CCT 推荐用户使用这两个文档类. 它们分别由 \LaTeX 2_ϵ 的标准文档类 `article` 和 `book` 导出, 用法也基本类似, 但排版格式上遵循中文的习惯. 使用时注意下述特点:

- 使用修改后的文档类, `book` 变为 `cctbook`, `article` 变为 `cctart`, 如

```
\documentclass[...]{cctbook}
```


不必再写 `\input chead.sty`.

- 只要不直接使用 `\zihao` 命令, 中西文字符总是同时变大或缩小, 可使用 `\small`、`\large` 等命令来同时改变中西文的大小.

12.2.3 Windows 下的 CCT

CCT 原来是伴随着 DOS 下的 emTeX 开发的, 到了 Windows 时代, DOS 版本的 emTeX 显得有点过时了, 它已于上世纪末期停止了更新, MiKTeX、fpTeX 等后来居上. 为了解决在 Windows 下的 TeX 系统中使用 CCT 的问题, 一些人开发了安装包, 将 CCT 移植到了 Windows 下的 TeX 系统中. 有兴趣的读者可到

<http://ctex.dhs.org/download/chinese.htm>

网址下载有关文件.

12.2.4 天元与 CCT 的互相转换

为了天元与 CCT 互相转换的需要, 在天元中已集成了转换软件. 用户在进入天元后点击‘Config’按钮, 选取‘CCT 与源文件’页, 在下半部有两个名为‘TY->CCT’与‘CCT->TY’的可选框, 根据需要选中其中之一, 退出 Config, 然后点击‘Open’按钮, 选中需转换的文件后, 就会自动按要求转换. 这两个转换软件都要求源文件中涉及到天元与 CCT 的命令连带它们的参数必须在同一行内, 中间不能换行. 例如 `\documentstyle{carticle}`, `\input tyinput`, `\input chead.sty`, `\zihao{5}` 等不能分成两行输入, 而且最好是经过编译证明无误的. 当然同一文本的天元与 CCT 版本的排版结果不可能完全相同, 但做到相似还是有可能的.

从天元转换成 CCT 时, 生成的 CCT 文件是以 `ctx` 为后缀的同名文件. 程序按如下对应关系修改:

<code>\半\五\六\七</code>	<code>\zihao{7}</code>	<code>\八\小</code>	<code>\zihao{6}</code>
<code>\九</code>	<code>\zihao{-5}</code>	<code>\标</code>	<code>\zihao{5}</code>
<code>\中</code>	<code>\zihao{-4}</code>	<code>\大</code>	<code>\zihao{4}</code>
<code>\特</code>	<code>\zihao{3}</code>	<code>\双</code>	<code>\zihao{2}</code>
<code>\巨</code>	<code>\zihao{1}</code>	<code>\叁\肆\伍\陆</code>	<code>\zihao{0}</code>
<code>\宋</code>	<code>\songti</code>	<code>\楷</code>	<code>\kaishu</code>
<code>\仿</code>	<code>\fangsong</code>	<code>\黑</code>	<code>\heiti</code>
<code>\题</code>	<code>\biaosong</code>	<code>\行</code>	<code>\ziti{F}</code>
<code>\粗</code>	<code>\ziti{G}</code>	<code>\姚</code>	<code>\ziti{H}</code>
<code>\美</code>	<code>\ziti{I}</code>	<code>\圆</code>	<code>\ziti{J}</code>

\准	\ziti{K}	\魏	\ziti{L}
\变	\ziti{M}	\隶	\ziti{N}
\琥	\ziti{O}	\综	\ziti{P}

其余的‘\汉字’的命令均被删去。英文命令 `\def\ChineseScale{????}` 保持不变, `\input tyinput` 改为 `\input cthead.sty`, `\input tdw` 或 `\input tdwc` 改为 `\def\draw#1,#2,#3{}`。当然 Tydraw 的作图功能没有了, 留下的是空白。

当把 CCT 源文件转换成天元源文件时, 生成的天元文件是以 `ty` 为后缀的同名文件, 并按如下对应关系修改:

\zihao{7}	\七	\zihao{6}	\小	\zihao{-5}	\九
\zihao{5}	\标	\zihao{-4}	\中	\zihao{4}	\大
\zihao{3}	\特	\zihao{2}	\双	\zihao{1}	\巨
\zihao{0}	\叁	\songti	\宋	\kaishu	\楷
\fangsong	\仿	\heiti	\黑	\biaosong	\题
\ziti{F}	\行	\ziti{G}	\粗	\ziti{H}	\姚
\ziti{I}	\美	\ziti{J}	\圆	\ziti{K}	\准
\ziti{L}	\魏	\ziti{M}	\变	\ziti{N}	\隶
\ziti{O}	\琥	\ziti{P}	\综		

其余的命令如 `\ziju{??}`, `\input cthead.sty` 均被删去, `{carticle}` 被改为 `{article}`。

在生成的天元文件的头部被插入以下 5 行:

```
\def\ChineseScale{1000}
\input tyinput
\newdimen\ccwd\newdimen\ccht\newdimen\ccdP
\setbox0=\hbox{大}
\ccwd\wd0\ccht\ht0\ccdP\dp0
```

由于 CCT 版本的改变, 上述转换不一定很完全。读者如有需要, 可根据你使用的版本与作者联系定制。

第十三章 输出到屏幕、投影仪或互联网

§13.1 如何显示彩色

为了在屏幕上显示彩色或打印彩色页面, 先要用

```
\usepackage[可选项]{color}
```

输入彩色宏包. 其中的可选项除了 §7.5.3 列举的驱动程序名外, 还可使用:

<code>dvipsnames</code>	使 <code>dvips</code> 的 <code>named</code> 彩色模式也适用于其它设备
<code>nodvipsnames</code>	关闭 <code>dvips</code> 的 <code>named</code> 模式, 以节省内存
<code>usenames</code>	使所有 <code>named</code> 色彩成为已经定义的

为了定义一种色彩, 可以采用如下的形式:

```
[模式]{数据}
```

可用的模式为: `rgb` (红, 绿, 蓝), `cmyk` (青, 洋红, 黄, 黑), `gray` (灰) 以及 `named` (已命名的). 数据是一组用逗号分隔的 0 与 1 之间的十进小数串, 表示每个分量的强度. 例如 `[rgb]{1,0,0}` 定义了红色, `[cmyk]{0,0,1,0}` 定义了黄色, 模式 `gray` 只取一个参数, 而 `named` 模式的数据则为驱动程序认识的色彩名, 例如 `[named]{Sepia}`. 例 13-1-1.tex 列举了 `dvips` 的 `named` 彩色模式, 经编译后能显示或打印出一个调色板, 把其中的 `dvips` 改成别的驱动程序, 即能得到那个驱动程序的 `named` 模式.

定义色彩的命令为:

```
\definecolor{色彩名}{模式}{数据}
```

它给色彩名指定一种颜色, 在以后的命令里可以使用这种色彩名. 在所有的彩色驱动程序中都被预定义的色彩名有: `red`, `green`, `blue`, `yellow`, `cyan`, `magenta`, `black`, `white`.

以下命令中的色彩可以是已定义的色彩名或者 '[模式]{数据}' 的形式, 例如: `{blue}`, `[rgb]{0,1,0}`.

`\pagecolor` 色彩

设定从本页开始的背景色, 以后的页都使用此背景色. 直至再遇到命令`\pagecolor`为止.

`\color` 色彩

选定某种色彩, 以后的文本就用这种颜色打印, 它的作用范围持续到所在环境的结束或者遇到另一条`\color`命令为止.

`\textcolor` 色彩{文本}

用指定的色彩打印文本.

`\colorbox` 色彩{文本}

生成一个以文本为内容的 LR 盒子, 并以指定的色彩作为盒子的背景色.

`\fcolorbox` 色彩一 色彩二{文本}

文本生成一个以文本为内容的带框 LR 盒子, 以色彩一作为框线颜色, 色彩二作为背景色. 不过这两种色彩必须都使用已定义的色彩名, 或者使用同一种彩色模式.

`\normalcolor`

把文字的颜色重置为前言末尾所激活的颜色, 通常是黑色. 前言中的`\color`命令可以改变‘标准’色彩.

比较好的工作方式是把要使用的色彩都用`\definecolor`定义好, 正文中只使用已定义的色彩名, 这样可以根据打印结果再作调整. 请注意同一种色彩使用不同的打印机会得到很不相同的效果, 屏幕显示与打印结果也有很大差距, 因此微调是必不可少的. 请读者试试编译及显示例 13-1-2.ty, 注意这个例子中使用了 Tydraw 的图形, 因此必须先用天元预处理. 这个例子的彩色要经 dvips 转换成 ps 文件后才能得到正确显示.

§13.2 如何使用 pdfTeX

使用 dvipdfm 可以直接把 dvi 文件转换成 pdf 文件, 不过它不会增加 L^AT_EX 中没有的 pdf 的功能. 因此了解 pdfTeX 的功能与用法是有好处的.

13.2.1 pdfL^AT_EX

pdfT_EX可用于处理Plain T_EX以及 $\mathcal{A}\mathcal{M}\mathcal{S}$ -T_EX文件,而pdfL^AT_EX则可直接处理L^AT_EX文件.一般的L^AT_EX源文件经pdfL^AT_EX处理后就得到了相应的pdf文件,但是带彩色的或利用graphics宏包嵌入图形的源文件必须加以修改,因为pdfL^AT_EX不能直接处理eps格式的图形文件.

利用本书光盘中所含的安装程序安装的WinEdt或WinShell集成环境都已经在工具条上建立了pdfL^AT_EX按钮,用鼠标点击即可.

使用彩色与嵌入图形

如果要使用彩色与嵌入图形,则应选取pdftex作为彩色驱动程序,而且应改用宏包graphicx,即在前言中声明

```
\usepackage[pdftex]{graphicx,color}
```

这样就能重新得到例13-1-2.ty的彩色表(参见13-2-1.tex以及13-2-2.ty).

pdfL^AT_EX中能使用的图形格式有: pdf, png, tif(tiff), jpg(jpeg),但不支持eps,为了在pdfL^AT_EX中使用eps图形,可先调用epstopdf.exe把它转换成pdf格式.

graphicx宏包的嵌入命令是:

```
\includegraphics[键 = 值, ...]{图形文件名}
```

有些键的值是整数,有的键值则为逻辑值,即true或false,对这样的键,单列键名而不输入键值相当于输入真值.键与键之间用逗号分开.可用的键有(注意有些键仅对eps图形有效):

scale= 数: 放大倍数;

width= 长度: 输出图形的宽度,如果height不给出,则按同一比例放缩;

height= 长度: 输出图形的高度,如果width不给出,则按同一比例放缩;

totalheight= 长度: 输出图形的总高度(高度加深度).如果要旋转图形,则应该用它取代height;

keepaspectratio (=true/false): 如果同时指定了width与height,那么这个标志保证原始图形的高宽之比不变,同时又使变换后的图形不超出指定的宽与高;

angle= 数: 作逆时针旋转的角度(以度为单位),它与键width与height的出现顺序会影响最终得到的图形(参见例13-2-3.tex);

origin= 位置: 规定旋转的中心,默认值是bl,表示左下角,也可取: c(中心), t(顶部), r(右端), B(基线),以及它们的组合;

draft (=true/false): 不插入图形, 仅保留出图形所占位置, 显示图形边界方框及图形文件名. 这可明显加快处理速度;

clip (=true/false): 截去边框外面的图形 (仅适用于 eps 图形);

bb= 左下角横坐标 左下角纵坐标 右上角横坐标 右上角纵坐标: 规定图形边框的坐标, 4 个长度间用空格隔开, 可以带有单位, 默认单位是大点 (bp) (仅适用于 eps 图形);

viewport= 左下角横坐标 左下角纵坐标 右上角横坐标 右上角纵坐标: 以图形的左下角作为坐标原点, 指定图形的边框. 与 clip 连用可截取图形的一个部分 (仅适用于 eps 图形);

trim= 左下角横坐标差值 左下角纵坐标差值 右上角横坐标差值 右上角纵坐标差值: 指定图形向内截去的值 (仅适用于 eps 图形);

hiresbb (=true/false): 在插入的 eps 图形文件中不使用通常的 %%BoundingBox 的值, 而是用 %%HiresBoundingBox 的值作为图边界盒子的值 (仅适用于 eps 图形).

graphicx 宏包的旋转盒子的命令是:

```
\rotatebox[键 = 值, ...]{角度}{文本}
```

可使用的键 **origin** 同上所示, 另一种指定旋转中心的方式是: $x = \text{长度}$, $y = \text{长度}$.

上述 graphicx 宏包在 pdfL^AT_EX 中的应用示例可见 13-2-3.tex.

对文字或图形作变换

还可以利用 pdf 的原始命令对图形作仿射变换, 这需要使用 pdfT_EX 的特有命令, 这些命令都是以 \pdf 起头的. 如下面的变换命令

```
\pdfliteral{q a b c d e f cm}%
...
\pdfliteral{Q}
```

其中 a, b, c, d, e, f 是 6 个实数, 它们刻画了一个仿射变换, 其长度单位是 bp:

$$\begin{cases} x' = ax + cy + e \\ y' = bx + dy + f \end{cases} \quad \text{或} \quad (x' \ y') = (x \ y) \begin{pmatrix} a & b \\ c & d \end{pmatrix} + (e \ f)$$

这里的 'q', 'cm', 'Q' 都是 pdf 的命令. 因此逆时针旋转 t° 的变换可描述成 $a = \cos t$, $b = \sin t$, $c = -\sin t$, $d = \cos t$, $e = f = 0$ (三角函数值应化成十进小数输入). 此外被变换的图形或文字应该放在一个长宽高都被设成 0 的盒子里, 在有关的命令之间不能有空格插入, 因此每条命令的后面要用注解号 '%' 除去后面

的空格. 为了使后继的内容有正确的留空, 必须作手工调整. 如下面的例子(参见 13-2-3.tex):

```
\newsavebox{\testbox}
\sbox{\testbox}{\Huge Rotated text box.}
\ht\testbox=0pt \wd\testbox=0pt \dp\testbox=0pt
This is a \pdfliteral{q 0.866 -0.5 0.5 0.866 0 0 cm}%
\usebox{\testbox}%
\pdfliteral{Q}
```

也可以用

```
\sbox{\testbox}{\includegraphics{pic.jpg}}
```

形式的命令对图形作变换.

其它图形文件

Xy-pic 宏包可以直接在 pdfL^AT_EX 下运行, T_EXdraw 则不行. MetaPost 生成的文件可用以下命令输入(在前言中要有 \usepackage[pdf_TE_X]{graphicx}):

```
\convertMPtoPDF{文件名}{横向放大倍数}{纵向放大倍数}
```

这里的文件名必须是运行 mpost 后生成的. 例如 13-2-3.tex 中的命令

```
\convertMPtoPDF{fig.1}{1.5}{0.9}.
```

插入注记

用 \pdfannot 命令可以建立一个注记, 下面是一个实例(参见例 13-2-4.tex):

```
\pdfannot width 10cm height 0cm depth 4cm
{
  /Subtype /Text      % 文本性质的注解
  % /Open true        % 如果取消注解号 '%', 则注解窗口是打开的
  /Contents(This is ...) % 注解内容, 其中不能有汉字
}
```

其中的尺寸是可选的. 也可插入声音或动画文件作为注记(参见例 13-2-4.tex):

```
\pdfannot width 0in height 0in depth 0in
{
  /Subtype /Sound
  /Sound << /F (SoundFile.wav) >>}
```

```
\pdfannot width 4in height 0in depth 3in
{
  /Subtype /Movie
  /Movie << /F (MovieFile.mov) >>}
```

建立链接

以下命令可建立链接的目标, 请注意 pdf 的链接是指向一个页, 不再深入到具体的位置.

```
\pdfdest 目标说明
```

目标说明的第一部分是指定一个用作标识的正整数或名字, 格式是 `num` 数字或 `name{标识}`; 第二部分是目标页显示的格式, 其选项如下表所示:

<code>fit</code>	窗口里显示整页
<code>fith</code>	使页宽适应窗口
<code>fitv</code>	使页高适应窗口
<code>xyz</code>	到当前位置, 后面可接 <code>zoom</code> 放大因子, 放大因子的值是整数, 相当于真实因子的 1000 倍

例如 (参见 13-2-4.tex):

```
\pdfdest name{dest1} xyz zoom 1500
```

建立一个链接可使用以下命令:

```
\pdfstartlink [尺寸] [属性] 动作说明
.....
\pdfendlink
```

其中的可选项尺寸具有 `width ... height ... depth ...` 的形式, 其默认值是包含在 `\pdfstartlink` 与 `\pdfendlink` 之间的内容的范围, 可以超过一页; 可选项属性说明边框的颜色与厚度, 例如

```
attr{/C [0.9 0 0] /Border [0 0 2]}
```

规定边框为暗红色; 动作可以是 `goto` 或 `user`, 如果是 `goto`, 后面应指明 `num` 数字, `name{标识}` 或 `page 页码{/Fit}`, 如果想链接到另一个 pdf 文件, 可使用 `goto file{文件名}`; 用户定义的动作可参见下例:

```
user{/Subtype /Link
  /A << /Type /Action
    /S /URI
    /URI (http://www.tug.org/) >>}
}
```

下面是链接的一个例子(参见 13-2-4.tex):

```
This is a link to destination
\pdfstartlink height 15pt depth 5pt
  attr{/C [0.9 0 0] /Border [0 0 2]}
  goto name{dest1}
'name\{dest1\}'
\pdfendlink
```

更详细的用法说明可参看 `texmf\doc\pdftex\pdftex-a.pdf`.

13.2.2 生成适合屏幕阅读的 pdf 文件

`pdfscreen` 是 C. V. Radhakrishnan 开发的, 它利用 `pdfTeX` 生成适合于屏幕上阅读的材料, 同时又能按正常的大小用 `dvi` 打印输出. 这一切都由选项 `print` 或 `screen` 控制. 在源文件里可以使用

```
\begin{print}...\end{print}
或
\begin{screen}...\end{screen}
```

形式的命令把只用于打印或屏幕显示的内容包裹起来.

根据作者的试验, `pdfscreen` 与 `CJK` 不完全兼容, 但天元能正常运行.

首先要读入宏包:

```
\usepackage[screen,panelleft]{pdfscreen}
```

而且这个命令最好放在导言的末尾, 以防止其他命令改变它的设置. `pdfscreen` 会自动输入宏包 `graphicx` 与 `color`. 方括号里的可选项有以下几种:

<code>screen</code>	产生适合屏幕输出的版本
<code>print</code>	产生适合打印输出的版本, 与 <code>dvi</code> 一样
<code>panelleft</code>	导航面板居左
<code>panelright</code>	导航面板居右
<code>nopanel</code>	取消导航面板
<code>paneltoc</code>	把目录放在面板上, 但只要正文中出现 <code>\tableofcontents</code> , 本命令即失效.
<code>sectionbreak</code>	在每一节的前面分页

导航面板与按钮有 6 种配色方案可供使用: `bluelace` (浅蓝), `blue` (蓝色), `gray` (灰色), `orange` (桔黄), `palegreen` (浅绿) 与 `chocolate` (浅棕色), 默认的是蓝色.

以下命令应出现在前言中:

`\screensize{高度}{宽度}` 规定屏幕尺寸, 用户可以自由取值, 但必须出现.

`\margins{左边}{右边}{顶部}{底部}` 规定页边宽度, 必须出现.

`\overlay{图形文件}` 选取背景图形.

`\overlayempty` 无背景图形.

`\paneloverlay{图形文件}` 选取导航板的背景图形.

`\paneloverlayempty` 导航板无背景图形.

`\changeoverlay` 出现本声明后, 每出现一个新节, 它的背景图形都会自动改变, 也就是改变颜色, 变化范围从 `overlay0.pdf` 至 `overlay10.pdf`, 共 11 种, 循环使用.

`\urlid{URL 地址}` 导航板上 'Home Page' 按钮指向的 URL 地址.

导航板上的按钮名称有多种语言可供选择, 可惜没有中文, 因此我们只能在前言里将它们重新定义, 具体方法请参看例 13-2-5.ty.

定义新的按钮可使用以下命令:

```
\addButton{按钮宽度}{名称}
\imageButton{宽度}{高度}{图形文件名}
```

后者是以图形作为按钮, 但是为使按钮发生作用, 必须把它包含在某个动作里. 例如:

`\Acrobatmenu{动作}{\addButton 或 \imageButton...}` 点击此按钮就会启动一个 Acrobat 动作, 这里的动作可以是: `FirstPage` (跳转第一页)、`PrevPage` (跳

转前一页)、NextPage(跳转下一页)、LastPage(跳转最后一页)、GoBack(返回)、GoToPage(跳转指定的页)、FullScreen(全屏显示)、Close(关闭本文件)、Quit(退出 Acrobat).

`\href{http://网址}{\addButton 或 \imageButton...}` 链接到指定的网址.

`\hyperlink{标识}{\addButton 或 \imageButton...}` 跳转到标识处, 标识可用 `\pdfdest named` 定义.

还有4个命令:

<code>\bottombuttons</code>	<code>\nobottombuttons</code>
<code>\topbuttons</code>	<code>\notopbuttons</code>

它们用来控制一个放在页底或页顶的小导航板, 依次表示设置或不设置页底导航板, 以及设置或不设置页顶导航板, 页底与页顶可以同时设置, 并不冲突. 它们的作用从出现命令的这一页开始, 一直延续到遇到相反的命令为止, 也可出现在前言中.

详细用法说明可参看 `texmf\doc\latex\pdfscreen>manual-screen.pdf`.

13.2.3 生成适合投影仪的 pdf 文件

`pdfslide` 也是 C. V. Radhakrishnan 开发的, 它利用 `pdfTeX` 生成适合外接投影仪使用的文件. 可是根据作者的试验, 在 `pdfslide` 环境里的 `\section` 命令与 CJK 不兼容, 而天元则可正常运行.

为了适应投影仪的特点, `pdfslide` 对字体作了放大, 它使用的 `\normalsize` 的字体大小相当于 16 pt, 因此为使汉字与西文匹配, 必须把天元的放大倍数设置为 1600 左右(可通过试验选取一个你认为合适的值), 参见例 13-2-6.ty. 而希望使用‘正常’大小的用户也可使用冠以 `\real...` 的系列命令, 例如: `\realnormalsize` (相当于 12 pt), `\reallarge` (相当于 14 pt) 等. 这时不要忘了改变天元的 Config 的 Magnification 为 1200!

`pdfslide` 的另一个特点是可以利用 Acrobat 的页面过渡效果, 利用命令

<code>\pagedissolve{选项}</code>

可以产生各种过场效果. 下表列出了一些可以使用的效果:

效果名	解 释
Split	两条线扫过屏幕展开新页, 类似于剧场的开幕或闭幕.
Blinds	类似于 Split, 但是有多条线同时进行, 好像打开百叶窗.
Box	通过从中心开始不断扩大的矩形展示新页.
Wipe	单独一根线扫过页面展开新页.
Dissolve	旧页‘溶化’, 产生新页.
Glitter	类似于 Dissolve, 不过效果从一边扫到另一边.
R	简单地以新页取代旧页, 无特殊效果, 这是默认值.

对某些效果还可以加上参数, 如下表所示:

键	解 释
/D	效果持续时间, 单位是秒 (适用于所有效果).
/Di 角度	指示 Wipe 或 Glitter 的运动方向, 角度应是 90 的倍数, 按逆时针方向计算, 0 表示自左向右.
/Dm	可能的值为 /H 或 /V, 分别表示水平或垂直的效果 (适用于 Split 及 Blinds).
/M	/O 表示效果从中心向边缘扩展, /I 则与之相反 (适用于 Split 及 Box).

例如:

```
\pagedissolve{Wipe /D 1 /Di 90}
\pagedissolve{Glitter /D 3 /Di 180}
\pagedissolve{Split /D 2 /Dm /H /M /O}
```

更详细的用法说明可参看 texmf\doc\pdfslide>manual.pdf.

§13.3 用 slides 制作投影片

L^AT_EX 的 slides 类是专门用于制作投影片的, 这是因为投影片有它的特点, 它的字体应该大些, 矮胖些, 而且不在段落中间换页, 并且往往需要彩色. 因此 slides 类使用无衬线字体, 而且 \normalsize 字体相当于 20pt, 如果使用天元, 应该把放大倍数设置成 1800 (供参考). 例如前言部分可以输入:

```

\def\ChineseScale{1800}
\input tyinput
\documentclass{slides}
\usepackage[dvips]{color}
.....
\begin{document}
.....
\end{document}

```

如果不使用天元, 则前面两行应该删去.

`slides` 里有3种环境: `slide`, `overlay`, `note`. 它们分别用以下方式进入:

```

\begin{slide} 文本及命令\end{slide}
\begin{overlay} 文本及命令\end{overlay}
\begin{note} 文本及命令\end{note}

```

`slide` 环境是 `slides` 的主体, 整个环境输出成一页, 并且有页码打印在下面, 如果超长, 则会发出警告. `overlay` 的内容是用于覆盖在 `slide` 打印的投影片上的, 因此可有不只一张, 而且应该接在相应的 `slide` 环境的后面, 它的页码附属于相应的投影片, 因此第6页投影片后面的覆盖片的页码是6-a, 6-b等等.

相应的投影片及覆盖片应该有相同的内容, 它们的区别就在于哪些部分被打印以及哪些部分不打印, 这是通过命令

```

\visible      \invisible

```

来实现的. 它们的作用规则与字体命令相同. 在投影片里, 只有少部分内容是不可见(`\invisible`)的, 而覆盖片则一开始就发出 `\invisible` 命令. 不过这两条命令仅适用于 `slides` 已定义的字体, 对于新增字体不起作用, 都是可见的, 因此不能使汉字‘隐身’. 不过办法还是有的, 就是把该隐藏的汉字统统改成全角空白(注意不能用两个半角空白代替), 使得它们虽占位但不显示, 例13-3-3.tex就是这样做的.

至于 `note` 则是用来提醒报告者的, 它的页码具有6-1, 6-2的形式.

在 `slides` 里也可使用 L^AT_EX 的命令

```

\pagestyle{格式}

```

其中的格式有下面3种:

plain 投影片、覆盖片以及注记片的页码都在右下角.

headings 同 **plain**, 只是当 **slides** 的 **clock** 选项被选中时, 注记片的左下角会打印出时间标记, 这也是默认的格式.

empty 不打印页码及对齐线.

`\pagestyle` 命令应该放在上述 3 个环境之外, 也可用 `\thispagestyle` 命令规定所在页的格式, 但作用范围只限一页.

当 `\documentclass` 命令中包含选项 **clock** 时, 以下两条命令就被激活了:

<code>\settime{秒数}</code>	<code>\addtime{秒数}</code>
---------------------------	---------------------------

其意义是自明的. 注记片上打印的是这些时间的累计, 而且以分为单位, 以提醒报告者.

当文件中含有许多投影片, 可在前言中利用以下命令选择其中一部分输出:

<code>\onlyslides{页码表}</code>	<code>\onlynotes{页码表}</code>
-------------------------------	------------------------------

前者输出指定页码范围的投影片、覆盖片以及注记片, 而后者仅输出指定页码范围的注记片, 页码表的形式如: 2,5,9-12,15, 必须由小至大顺序排列, 其中可以包含并不存在的页码.

最后我们看一个例子 (13-3-2.tex), 其输出如附图所示.

```

\documentclass[a4paper,clock]{slides}
\begin{document}
\begin{center}\Large\bfseries
Sample Viewgraphs
\end{center}

\begin{slide}
\begin{center}\large Advantages of \texttt{slides}\end{center}

\begin{itemize}
\item Uses special fonts
\item Forces key words instead of long text
\invisible
\item Supports color layers
\visible

```

Sample Viewgraphs

Advantages of slides

- Uses special fonts
- Forces key words instead of long text

Color commands may be used

1

- Supports color layers

as color layers

1-a

Add the overlay only for L^AT_EX 2.09

5 min

1-1


```
\end{itemize}
```

```
Color commands may be used {\invisible as color layers}
```

```
\end{slide}
```

```
\begin{overlay}
```

```
\invisible
```

```
\begin{center}\large Advantages of \texttt{slides}\end{center}
```

```
\begin{itemize}
```

```
\item Uses special fonts
```

```
\item Forces key words instead of long text
```

```
\visible
```

```
\item Supports color layers
```

```
\invisible
```

```
\end{itemize}
```

```
Color commands may be used {\visible as color layers}
```

```
\end{overlay}
```

```
\addtime{300}
```

```
\begin{note}
```

```
Add the overlay only for \LaTeX~2.09
```

```
\end{note}
```

```
\end{document}
```

§13.4 生成html文件

本节介绍如何使用 \TeX 4ht从 \LaTeX 源文件生成html文件. 本书所带光盘的安装程序有安装 \TeX 4ht的选项, 选中后就能自动安装. 不过含有简体汉字的 \LaTeX 源文件目前还不能转换成html文件.

我们先看一个例子(13-4-1.tex):

```
\documentclass[a4paper,12pt]{article}
```

```

\usepackage{colortbl}
\author{P. Anyname}
\title{Some questions about mathematics}
\begin{document}
\maketitle
\tableofcontents
\section{First Section}
\subsection{First Sub-section}
An equation:  $x^2+y^2=z^2$ .

\noindent Another equation:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial y}$$

\subsection{Second Sub-section}
A table:
\begin{tabular}
{|>{\columncolor{yellow}\bfseries}l|>{\columncolor{red}}p{4cm}|}
A&1\\\hline
B&2\\
\end{tabular}
\section{Second Section}
\section*{Third Section}
\end{document}

```

键入命令:

```
htlatex 13-4-1 "参数一" "参数二" "参数三"
```

就能生成文件 13-4-1.html. htlatex.bat 除了调用 L^AT_EX 外, 还调用了 tex4ht 以及 t4ht 两个程序. tex4ht 生成 html 文件, 而 t4ht 则生成数学公式所需的图形文件. 参数一是供 L^AT_EX 用的, 下面有介绍; 参数二是供 tex4ht 用的, 指示从 ht-fonts 的根目录到子目录的路径, 在一开始设定好后就不必再设置了; 参数三是供 t4ht 用的, 唯一可用的值是 -p, 它告诉 t4ht 不要生成图形, 当你已经生成了所需的图形文件, 以后只是修改文本或格式时, 选用此参数可以节省时间. 如果前两个参数为空, 则应使用 "" "" "-p" 的形式.

参数一的常用取值是:

html 生成 html 文件, 这是默认值.

- 2 生成的html按目录的二级标题(\section)分割成多个文件,这个选项的取值范围从1(按\chapter分割)直至4(按\subsubsection分割).没有此选项时,总是生成单独一个html文件.

section+ 建立从目录到相应章节的链接.

frames 使目录与正文分别放在两个框架内.

参数一的第一项一般是扩展名为`cfg`的配置文件名,其格式见后面说明,因此不使用配置文件时,第一项应为`html`,例如: "`html,2,sections+`".读者可选用各种不同参数编译`13-5-1.tex`,以观察其效果.

另一种方法是建立一个配置文件,它的扩展名必须是`cfg`,格式为:

```

前导定义
\Preamble{选项}
...前言定义...
\begin{document}
...头部定义...
\EndPreamble

```

其中的选项就是上面提到的参数一,例如`\Preamble{html,2,section+}`相当于`\usepackage[html,2,section+]{tex4ht}`.前导定义与前言定义都插在源文件的前言中,分别位于读入宏包`tex4ht`命令的前后,而头部定义则放在生成的html文件的头部.这些定义中可用到下列的`TeX4ht`的底层命令.使用配置文件的好处是可以构造出各种复杂格式的html文件,而不需更动`LaTeX`的源文件.我们这里附了两个配置文件: `htlatex1.cfg`与`htlatex2.cfg`,前者很简单,后者则建立了框架结构,试分别用这两个配置文件编译`13-4-1.tex`,并作比较.读者会发现使用框架结构时左边有两个Contents,只要把`13-4-1.tex`中的`\tableofcontents`注解掉就可解决.

TeX4ht的几个底层命令

`\HCode{...}` 直接插入html命令,例如`\HCode{<HR>}`(画一条水平直线).

`\HPage{锚名}内容\EndHPage{}` 建立一个超文本页,它可以通过锚名进入,含在内容中的命令`\ExitHPage{锚名}`会创立一个退出按钮.例如:

```

\HPage{Enter another page}
..... [\ExitHPage{Exit this page}] .....
\EndHPage{}

```

`\Link[目标文件]{目标位置}{当前位置}锚名\EndLink` 建立一个链接到‘目标文件#目标位置’的锚, 而且为当前位置建立标记. 当选项目标文件为空时, 默认值是当前文件所创建的文件. 特殊字符~, _, %应分别输入为: ‘\string ~’, ‘\string _’以及‘\%’. 例如:

```
\HPage{}
.....
\Link[http://www.tug.org/]{XX}\TeX{} Users Group Home Page
\EndLink
.....
\EndHPage{}
.....
\Link{XX}{...}\EndLink
```

`\ifHtml... \else... \fi` 根据html状态与非html状态分别加入不同内容, 当非html状态的内容为空时, 可省去`\else`. 例如:

```
\ifHtml \HCode{<HR>} \else \hrule \fi
```

与图形有关的命令

`\Picture[说明文字]{图形文件名}` 指向一个图形文件. 例如:

```
\Picture[**OSU logo**]{http://www.cis.ohio-state.edu/images/OSU.gif}
```

`\Picture+[说明文字]{文件名_属性}内容\EndPicture` 根据所给的内容作图, 并存储在一个图形文件中, 同时指向此图形文件. 当文件名为空时, 使用自动生成的文件名, 不过当后面有属性时, 第一个字符应是空格. 例如:

```
\Picture+[ align="right"] Text within a picture. \EndPicture
```

创建一个以文本为内容的图形.

`\Picture*[说明文字]{文件名_属性}内容\EndPicture` 与带+号的命令类似, 只是生成一个竖直的盒子.

我们把上面的一些命令加入后做了一个例13-4-2.tex, 请读者用不同配置编译比较.

关于TeX4ht的详细说明请参看`\texmf\doc\html\tex4ht\mn.html`.

附录一 TeX 系统的安装

这里将重点介绍 MiKTeX 2.1 版以及集成环境 WinEdt 与 WinShell 的安装方法. 对于机器配置较低的读者, 则可以安装 DOS 环境下 16 位的 emTeX, 为方便用户, 已将常用命令放在批处理程序文件中, 模拟成一个集成环境. 还介绍了使 PCTeX32 与天元配合的方案, 需要的文件都储存在光盘上. 不过后两种解决方案由于版本的限制, 不能保证本书的内容都能正确实现. 本书还提供了光盘版的 MiKTeX, 只要在硬盘上安装少量文件, 就能在光盘上运行 MiKTeX (当然速度较慢), 而且可以干净卸载, 特别适用于临时借用别人机器运行 MiKTeX 的用户.

A1.1 自动安装 MiKTeX

在光盘上有一个自动安装程序, 用户只要进入 \install 目录, 执行 setup.exe, 就能启动自动安装程序.

启动安装程序后, 首先出现一个安装说明窗口, 请读者仔细阅读安装说明, 读完后按 'Next' 进入确定安装路径的窗口. 请选定一个有足够剩余空间的硬盘, 为叙述方便, 以下假设是 D 盘, 并假定你的光盘是 F 盘. 那么就选定安装路径为 D 盘的根目录 D:\. 以后所有的软件都安装在这个 D 盘上. 选定后按 Next 进入下一个选择安装内容的窗口. 这个窗口列举了 13 个选项, 前面 5 个是独立的软件:

安装 MiKTeX 2.1 这是 Windows 环境中使用的 TeX 系统, 也是安装的核心. 选中后将启动 MiKTeX 本身的安装程序, 其详细过程参见第 241 页的说明. 注意你应该把 MiKTeX 安装在默认路径 d:\texmf 以及 d:\localtexmf 中. 为了使指向 MiKTeX 的可执行程序的路径生效, 必须在完成安装后重新启动计算机.

安装 WinEdt 这是方便运行 TeX 的一个集成环境, 相当好用, 不过是共享软件, 只能试用一个月. 请用典型安装的方式把它安装在默认路径 d:\program files \winedt 内, 并且不要选 Launch the program file.

安装 WinShell 这是另一个运行 TeX 的一个集成环境, 而且是自由软件, 没有任何限制, 只是功能不及 WinEdt 强大. 请将它安装在默认路径 d:\program

files\winshell 内. 这两个集成环境只要选一个就够了.

安装 Ghostscript 这是 ps 语言的解释程序, 它与下面的 GSview 一起提供了显示与打印 ps 文件的工具, 相当有用. 这里安装的是 Ghostscript 7.00 以及 GSview 4.0, 它们都不是自由软件, 不过你仍可以使用, 只是在每次启动时会显示一个提醒你注册的窗口. 这两个软件的安装路径分别取成默认的 d:\gs 以及 d:\Ghostgum.

安装 GSview 见上面一项的说明.

对于上述 5 个软件, 我们的安装程序只是启动它们各自的安装程序, 因此高级用户完全可以分别单独安装上述软件或它们的更新的版本. MiKTeX 的安装参见第 241 页的介绍, 其它几个软件的安装与配置可参见本附录的 A1.3 节 (注意必须安装在默认的路径里, 否则你必须自己修改配置文件).

再介绍后面的 8 个选项:

MiKTeX 补充: diagrams 包 在 9.16.2 节介绍了画交换画图的宏包 diagrams, MiKTeX 没有收录这个宏包, 因此你若想使用这个宏包的话, 必须补充安装.

例及测试文件 包括本书的一些例子以及几个可用于测试安装是否成功的小例子, 如果你选中这一项, 那么它们将被安装在 d:\texuser 目录内, 你也可以不安装, 而在需要时从光盘的 \examples 中复制.

安装天元 把天元执行程序安装在 d:\tywin 内, 同时也复制 Tydraw 所需的字库的 METAFONT 文件.

WinEdt 需用的天元补充文件 如果同时安装集成环境 WinEdt 以及天元, 需要对 WinEdt 原有的配置作修改, 用修改后的文件覆盖原有的.

WinShell 需用的天元补充文件 如果同时安装集成环境 WinShell 以及天元, 需要对 WinShell 原有的配置作修改, 用修改后的文件覆盖原有的.

CJK 补充文件 如果你想使用 CJK 宏包, 也想使用中文 Windows 带有的中文字库, 必须复制这些补充文件.

CJK 的 PS 字库 如果你想使用 CJK 宏包, 也想使用 dvips 以及 pdftex 处理汉字的话, 增加这些汉字的 ps 字库会加快处理速度, 提高输出质量. 不过不使用 ps 字库的预览程序 yap 及打印程序不会用到 ps 字库.

TeX4ht 的补充文件 如果你想使用 TeX4ht 生成 html 网页文件的话, 必须安装这些可执行文件与配置文件.

请用户根据自己的需要来选择 (当然 MiKTeX 是必选的):

1. 对于不使用汉字的用户, 你只需再选择一个集成环境, 并根据需要, 确定是否要安装 diagrams 宏包以及 TeX4ht. 对于选择 MiKTeX 最小安装的用户, 还需

要添加一些宏包, 才能编译本书中所有不含汉字的例子. 添加宏包的方法如下:
选中

开始 → 程序 → MiKTeX → MiKTeX Options

出现相应窗口后选取 'Packages' 页, 先在 'Download Site' 栏里选中你的光盘内的目录 `\miktex\tm\packages`, 左边小窗口会出现一个目录树, 在你需要添加的宏包目录前的小方块里用鼠标点击, 使得里面出现一个小勾, 就表明给选中了.

a. 如果你需要使用 CJK 包的, 则应选取

Languages → Chinese, Japanese, Korean → cjk

b. 如果你需要使用 TeX4ht 包的, 则应选取

SGML/XML/HTML typesetting → tex4ht

c. 如果你需要使用 pdfLaTeX 的, 则应选取

Formats → ConTeXt → pdfTeX

d. 如果你需要使用 MetaPost 包的, 则应选取

Formats → ConTeXt → MetaPost

e. 如果你需要使用画图用的 TeXdraw 与 xyPic 包的, 则应选取

Applications → Graphics → texdraw

Applications → Graphics → xypic

f. 如果你需要编译 pdfLaTeX 的例 13-2-*.*, 则还应选取

Formats → LaTeX → Basic LaTeX → carlisle

Formats → LaTeX → LaTeX packages → comment

Formats → LaTeX → LaTeX packages → fancybox

如果这些包都安装了, 那么 `\texuser` 里的所有例子都能编译了.

2. 如果想使用天元, 那么除了选中 '安装天元' 外, 还应根据你选择的集成环境, 选择安装补充文件.

3. 如果要使用 CJK 宏包, 则应选中 'CJK 补充文件', 并根据需要决定是否要选 'CJK 的 PS 字库'.

如果所有选项都选中, 并不会互相冲突, 只是多占些硬盘空间而已.

关于 MiKTeX 及其相关软件的安装与配置的更进一步介绍请参看本附录的 A1.3 节. 尤其是关于天元的运行及配置, 请没有经验的读者务必看一下本附录的 A1.3.4 与 A1.3.5 小节.

自动安装后运行不正常的手工修改

我们的自动安装程序有可能在某些特殊配置的操作系统下运转不正常, 可能出现的现象是: 天元或 TeX4ht 运行不正常, WinEdt 或 WinShell 中不出现天元的按钮等. 问题的根源是对一些配置文件的修改没有正确执行. 对这些文

件的修改方法十分简单: 除对天元的 Config 作检查外, 只要把下面指出的文件内的盘号 d:\ 全部替换成安装 MiKTeX 的硬盘的盘号即可. 最可能出问题的是 WinShell 的配置文件, 因为它们的位置与操作系统有关. 如果同时安装了天元和 WinShell, 则先对 \program files\winshell\winshell.ini 按上述方法作修改, 然后在你的机器里搜寻所有名为 winshell.ini, winshelltools.bmp 的文件, 用 \program files\winshell 目录里的修改好的同名文件一一代替即可. 如果你安装了 TeX4ht, 则需检查修改 \texmf\miktex\bin 目录中的文件 htlatex.bat, httx.bat, tex4ht.env; 如果同时安装了天元和 WinEdt, 则需修改 \program files\winedt 目录里的 winedt.ini.

A1.2 如何安装光盘 TeX

用户只要进入 \cdtexinst 目录, 执行 setup.exe, 就能启动自动安装程序.

启动安装程序后, 首先出现一个安装说明窗口, 请读者仔细阅读安装说明, 读完后按 'Next' 进入确定安装路径的窗口. 请读者选定一个工作硬盘, 假设是 D 盘, 就选定安装路径为 D 盘的根目录 D:\. 选定后按 Next 进入下一个选择安装内容的窗口, 其中包括 '安装光盘 TeX'、'安装 GSview 到硬盘上' 以及 '卸载光盘 TeX' 三个选项. 选取 '安装光盘 TeX' 后, 就会自动进行安装, 在结束前会打开 'MiKTeX Options' 窗口, 你应该做以下操作:

进入 'Roots' 页, 其中第一行的目录应该是 "硬盘号:\localtexmf", 它的 Description 是 "Local TEXMF dire...", 下面一行的目录是 "光盘号:\texmf". 如果不对, 则应删除后重新修改. 删除方法是: 用光标点击这一行使其处于选中状态, 再点击右边的 'Remove' 按钮, 就能把这一行清除. 把这两行都清除后, 点击 'Add' 按钮分别加入目录 D:\localtexmf 以及 F:\texmf (注意次序不能颠倒, \localtexmf 必须要在上面, 而且这里的 D 与 F 应该分别用你自己的硬盘号与光盘号代替), 用鼠标点击 D:\localtexmf, 使其成为选中状态, 利用 'Declare Local' 按钮将其设置成 'Local TEXMF Directory', 核对无误后点击 '应用' 让其执行.

再进入 'General' 页, 点击 'Refresh Now' 按钮以更新 FNDB (File Name Database). 完成后点击 '确定' 按钮退出.

这样就完成了光盘 TeX 的安装, 并且已经集成了天元、CJK 以及 TeX4ht. 以后只要执行命令 D:\localtexmf\bin\winshell 就能进入集成环境, 为方便起见可以在桌面上生成一个快捷方式. 安装时已经在硬盘上建立了一个 \texuser 目录, 里面有一些测试文件: small2e.tex 是测试 L^AT_EX 2_ε 的, 2-1-1.ty 是测试天元的, 2-1-1cjk.tex 是测试 CJK 的, 13-2-1.tex 需要用 pdfL^AT_EX 编译, 生成的是 pdf

文件, 13-4-1.tex 需要用 TeX4ht 编译, 生成 html 网页. 这些编译程序都可通过点击 WinShell 工具栏中的相应按钮启动执行.

不过对于 Windows2000 或 Windows ME 的用户来说, 如果你要使用天元, 必须先点击 WinShell 工具条上的‘天元 CFG’按钮, 进入: Config → 字模文件名, 把这一页里的 simsun.ttf 全部改为 simsun.ttc, 存盘退出后就能正确运行天元了. 另一方面请你试一下编译一个文件, 用 dvips 生成 ps 文件, 再点击工具条上的戴眼镜的幽灵以执行 GSview, 如能正确显示, 那么安装成功了, 否则, 说明在光盘上不能运行 GSview, 只好重新启动安装程序, 选择‘安装 GSview 到硬盘上’, 作补充安装.

要卸除光盘 TeX 也很容易, 只要执行 \cdtexinst\setup.exe, 选中工作硬盘的根目录, 选择‘卸载光盘 TeX’, 就能把有关的子目录与文件删除干净. 卸载时请别忘了把你自己在 \texuser 里存放的 TeX 或 ty 源文件另存下来. 此外, 如果你把 GSview 安装在硬盘里, 那么你必须自己把它们删除.

自动安装后运行不正常的手工修改

我们的自动安装程序有可能在某些特殊配置的操作系统下运转不正常, 可能出现的现象是: 天元或 TeX4ht 运行不正常, WinShell 中不出现天元的按钮等. 问题的根源是对一些配置文件的修改没有正确执行. 对这些文件的修改方法十分简单: 除对天元的 Config 作检查外, 要修改的文件都在 \localtexmf\bin 里, 只要把下面指出的文件内的盘号 d:\ 全部替换成工作硬盘的盘号, 并把盘号 f:\ 全部替换成光盘的盘号即可. 最可能出问题的是 WinShell 的配置文件, 因为它们的位置与操作系统有关. 先对 \localtexmf\bin\winshell.ini 按上述方法作修改, 然后在你的机器里搜寻所有名为 winshell.ini, winshelltools.bmp 的文件, 用 \localtexmf\bin 目录里的修改好的同名文件一一代替. 最后检查修改文件 htlatex.bat, httex.bat, tex4ht.env.

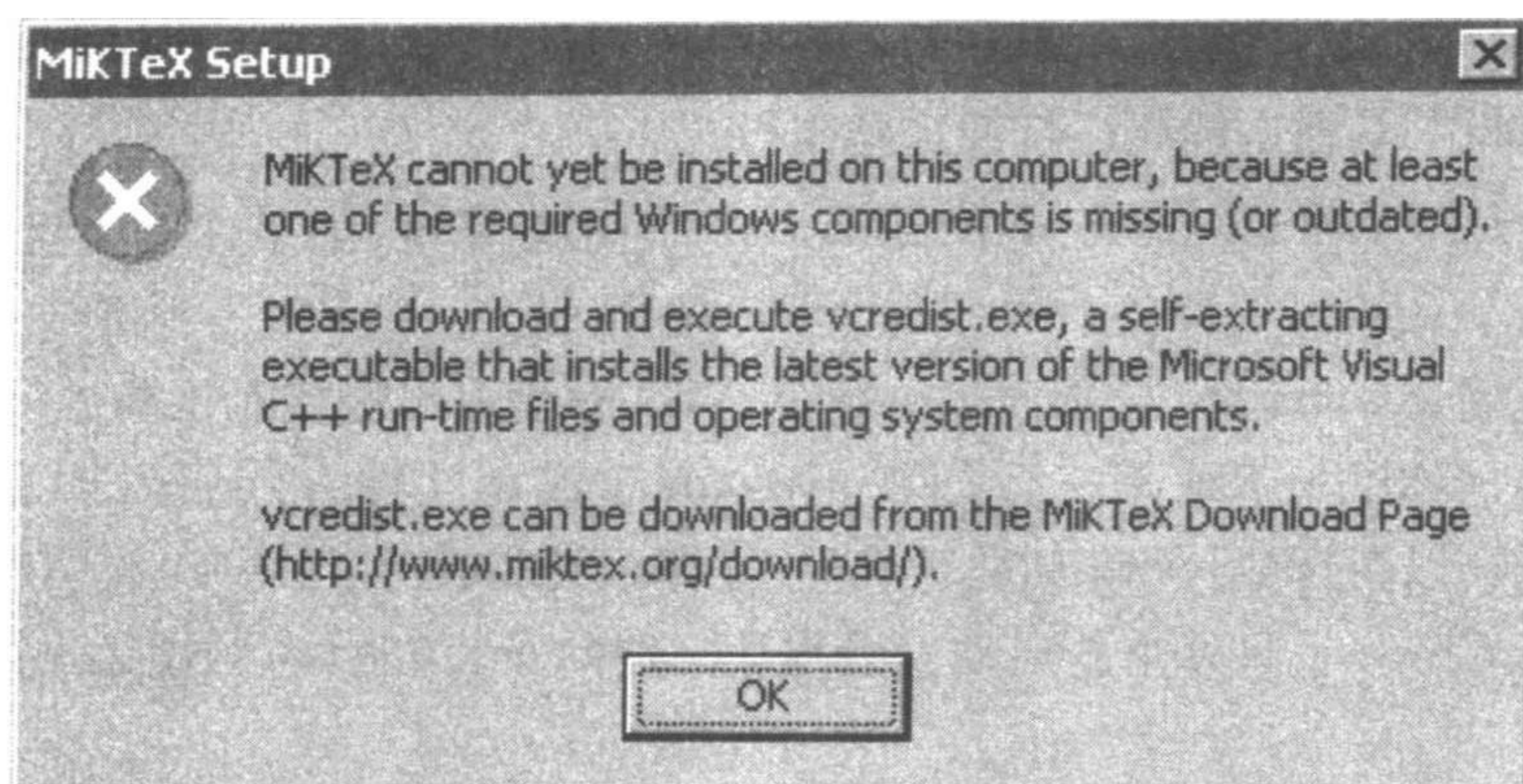
A1.3 MiKTeX 及其相关软件的安装与配置

A1.3.1 安装 MiKTeX

安装 MiKTeX 并不困难, 只要执行光盘中的文件

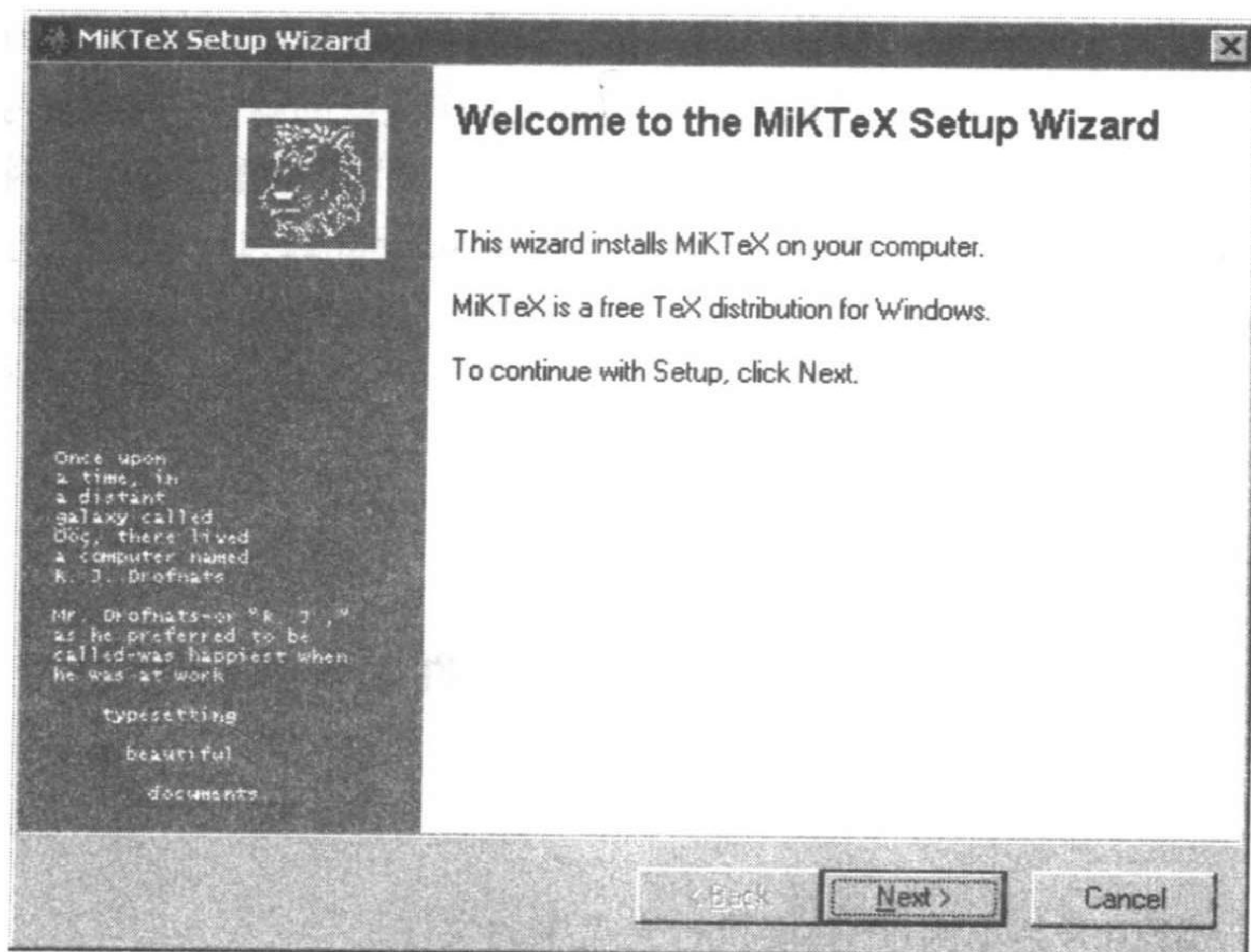
\miktex\setup\setup.exe

就进入自动安装环境, 如果你的操作系统缺少一些动态库文件, 会出现如下的警告信息:

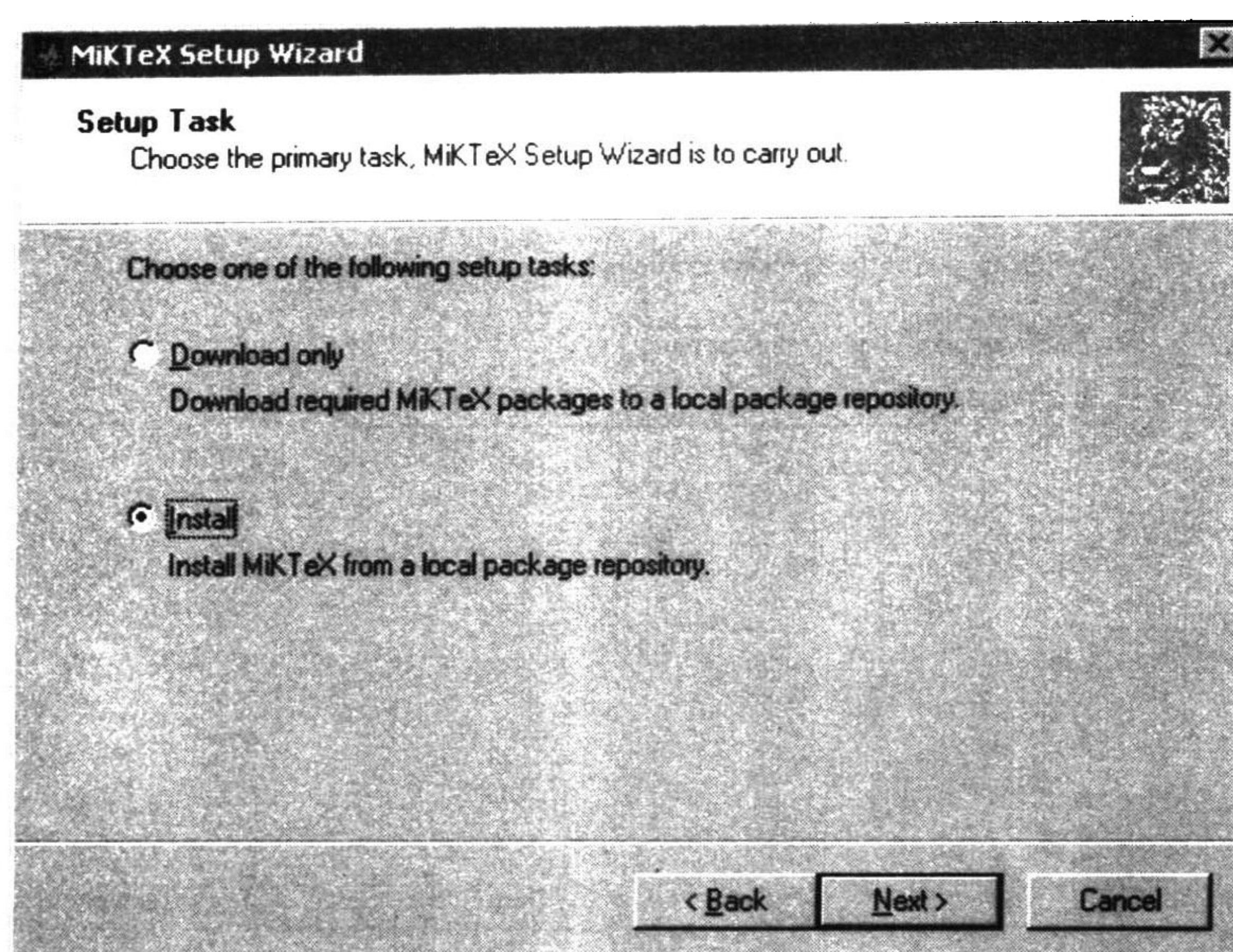


你应该按 OK 退出, 并把光盘中含于 `\windows\system` 目录内的 4 个文件复制到你的系统的相应目录中 (一般应为 `c:\windows\system`).

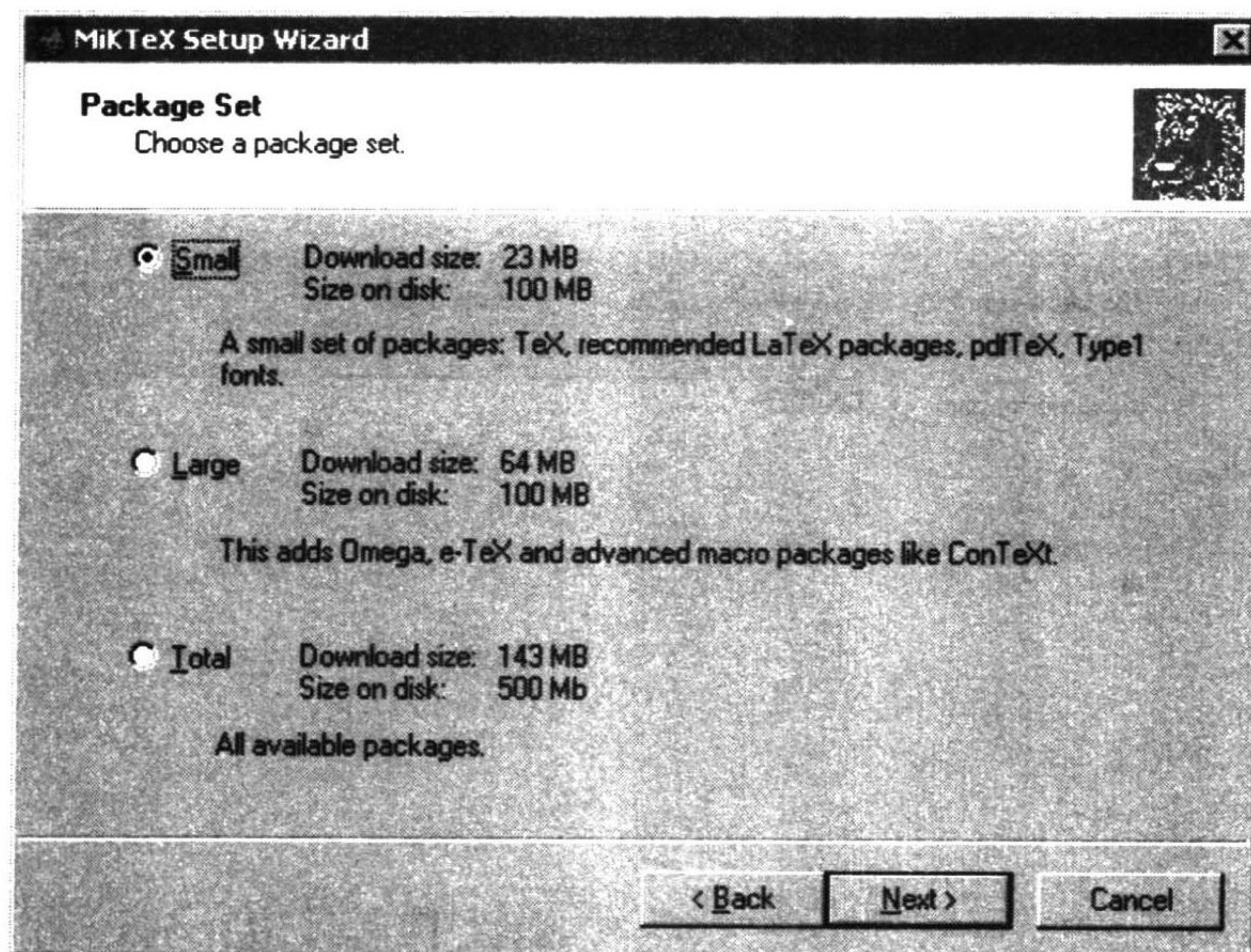
再次进入自动安装环境, 即可见到以下的欢迎窗口:



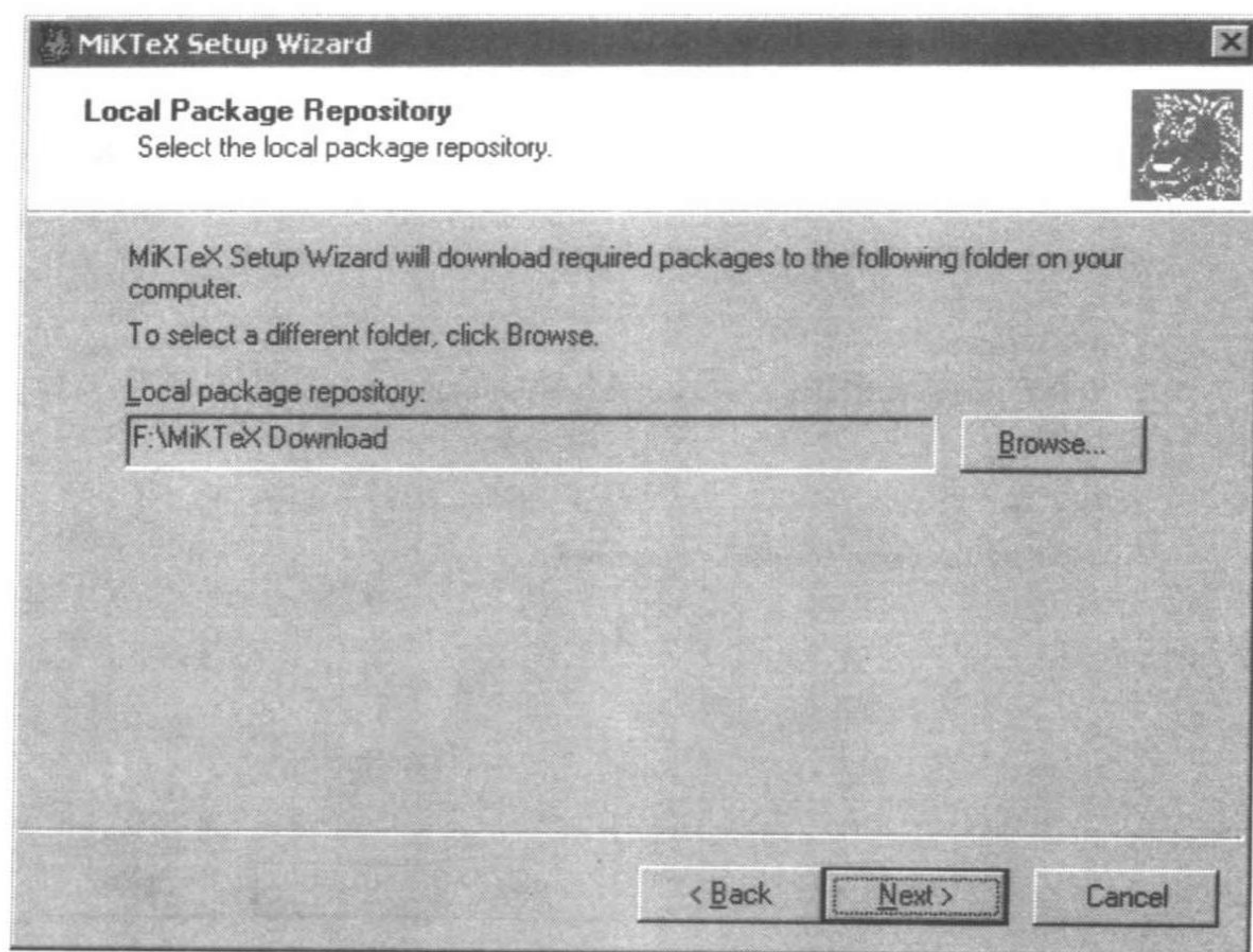
点击‘下一步’后, 出现以下窗口:



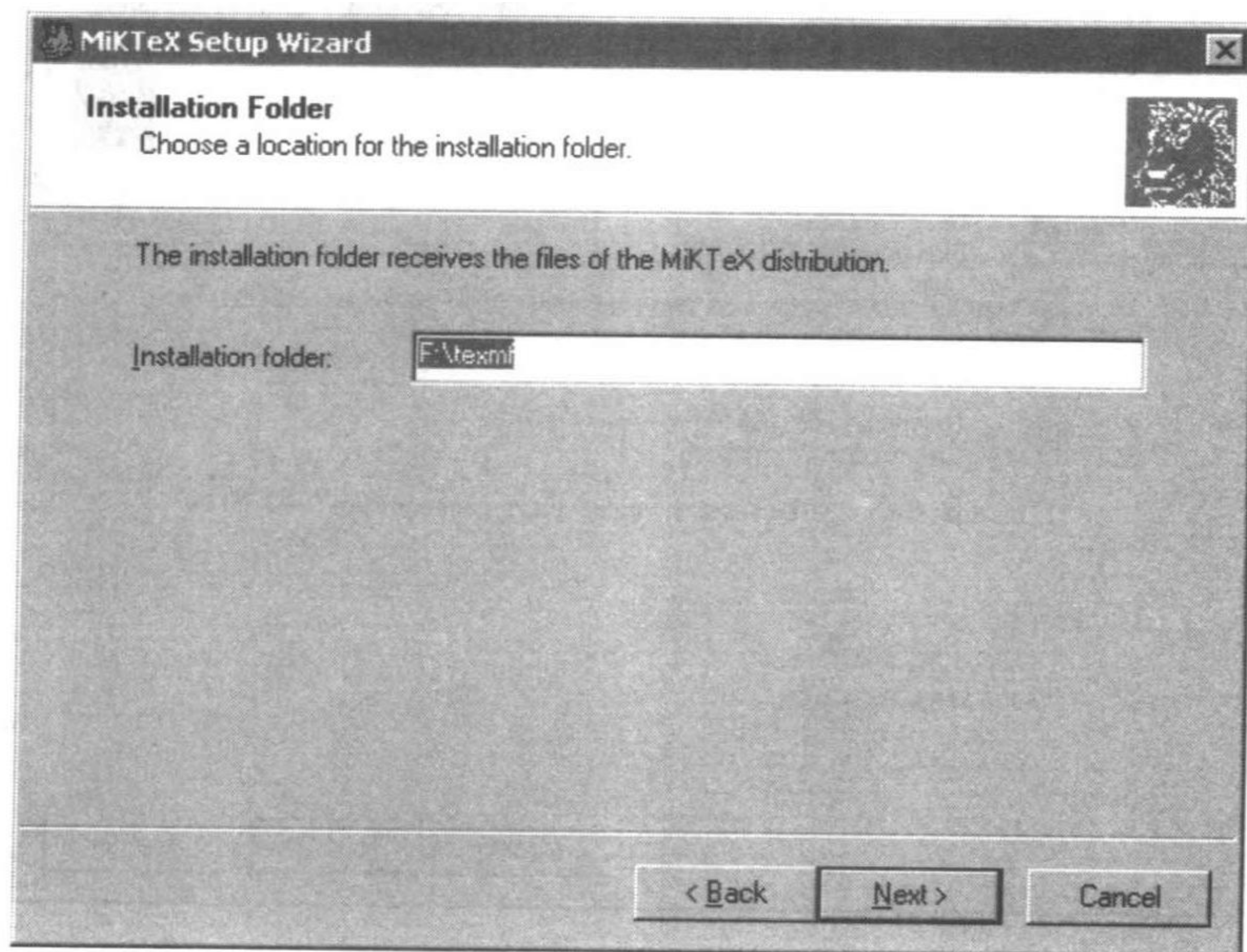
点击‘下一步’后, 出现以下窗口:



我们建议你先选择‘Small’, 以后可以根据你的需要, 再添加其他模块. 点击‘下一步’后, 出现以下窗口:

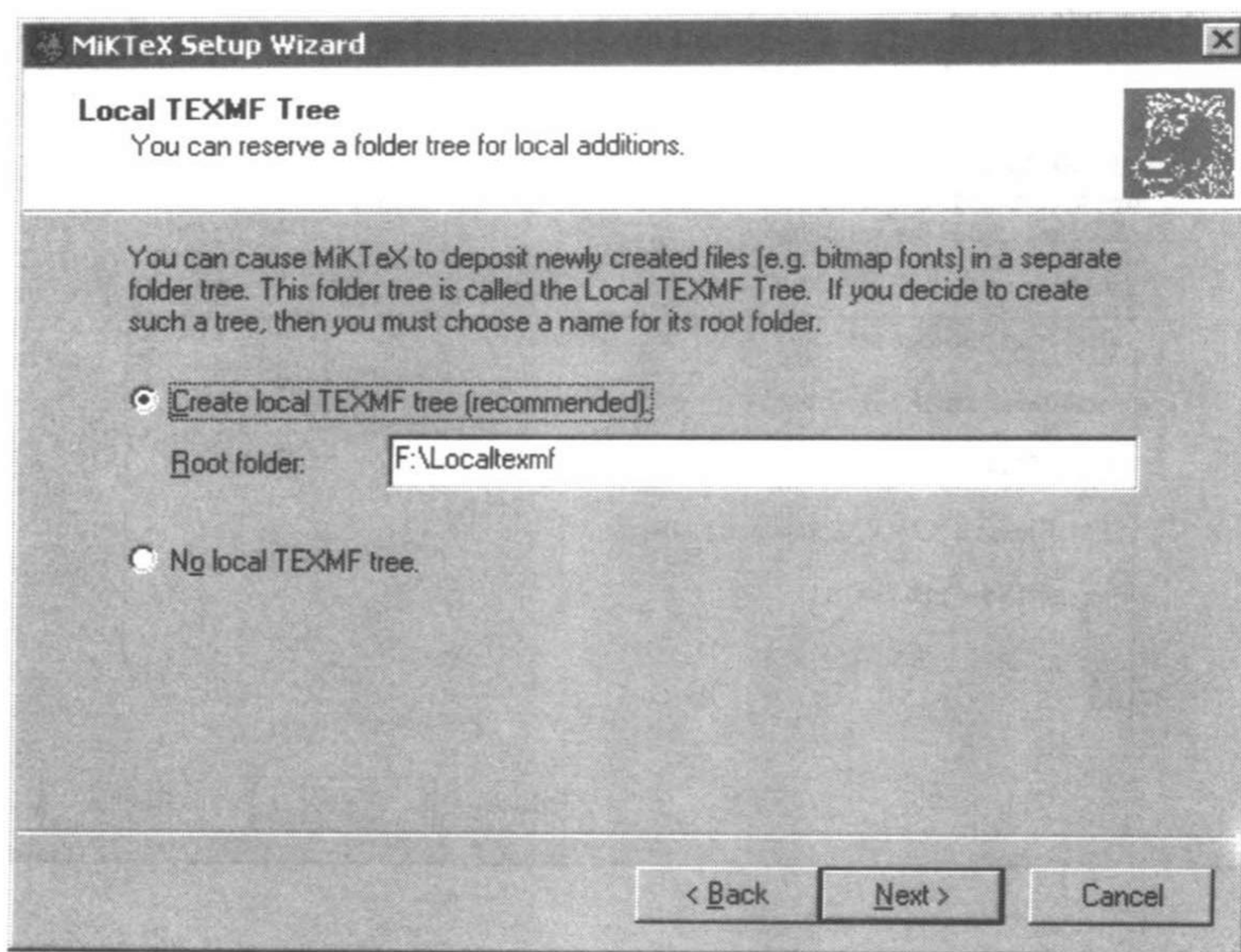


方框中显示的路径应该是 `F:\MIKTEX\pm\packages` (假定你的光盘驱动器的盘号是 F), 一般不需修改, 如觉得有问题, 可点击 'Browse' 选择正确的路径. 点击 '下一步' 后, 出现以下窗口:

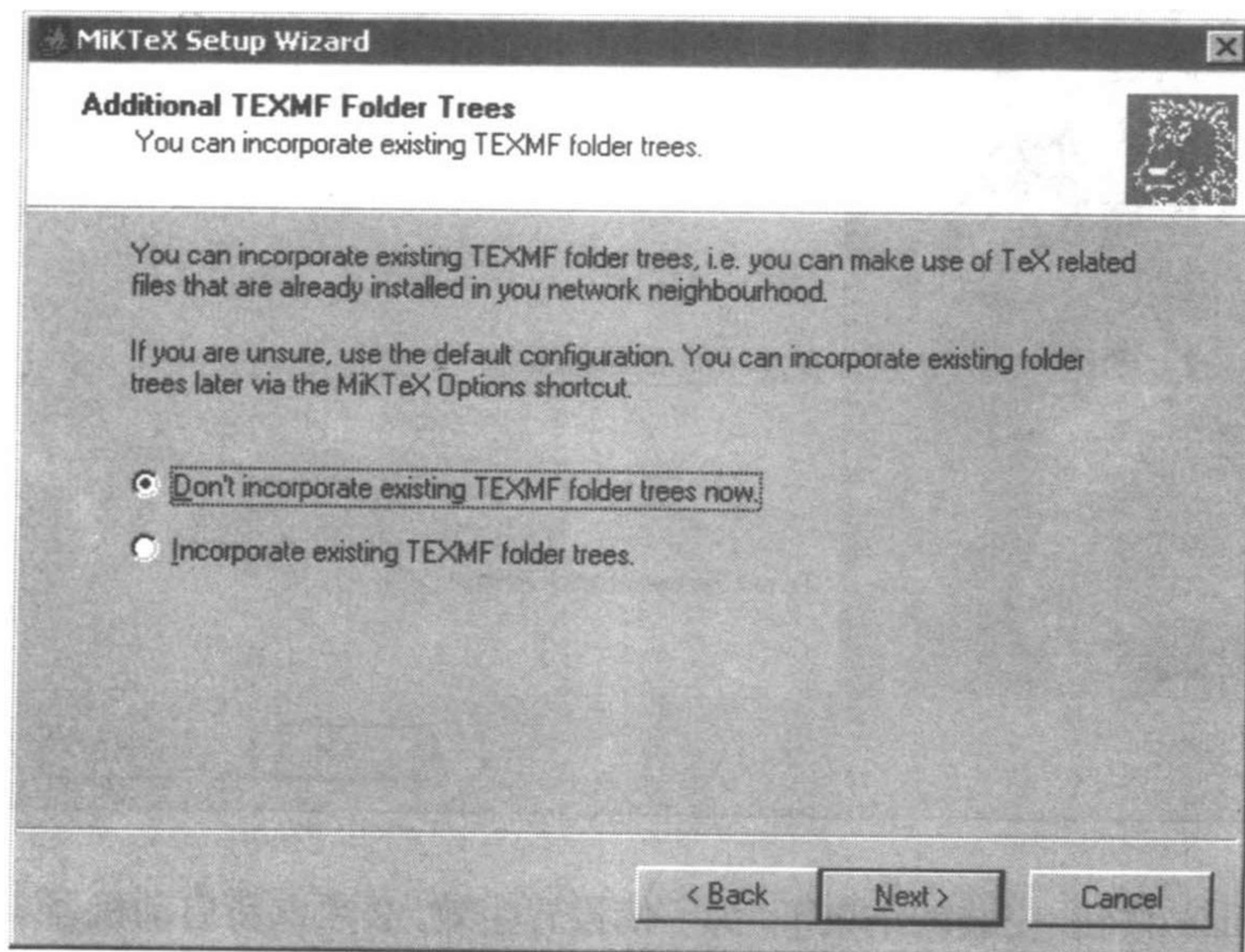


要你选择 MiKTeX 在你的机器上的安装路径, 方框中显示的路径是 `F:\texmf`, 建议你只改变盘号, 例如把 'F' 改为 'D', 采用默认的路径名. 点击 '下一步' 后, 出现

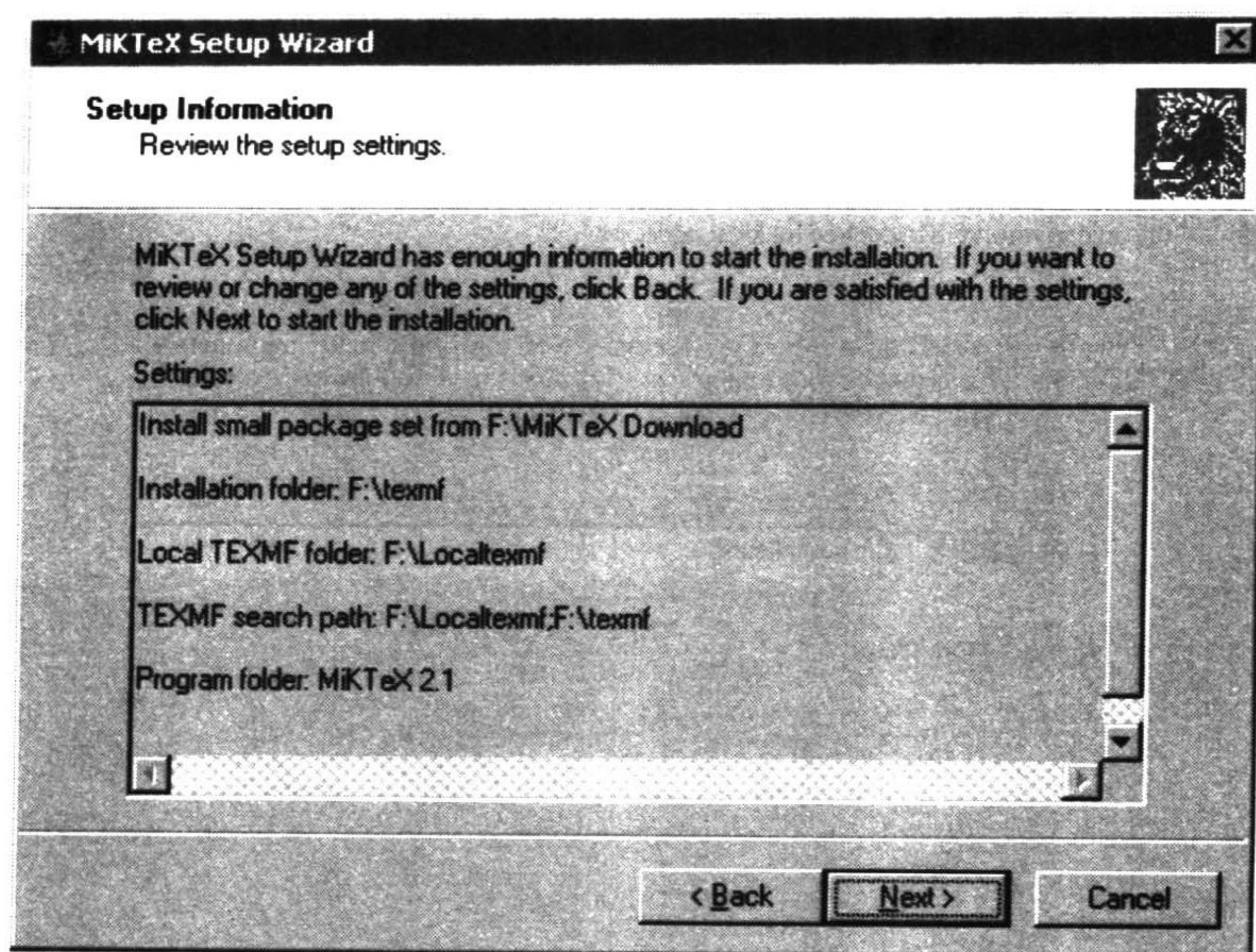
名为‘Program Folder’的窗口, 可采用其默认值, 点击‘下一步’后, 出现以下窗口:



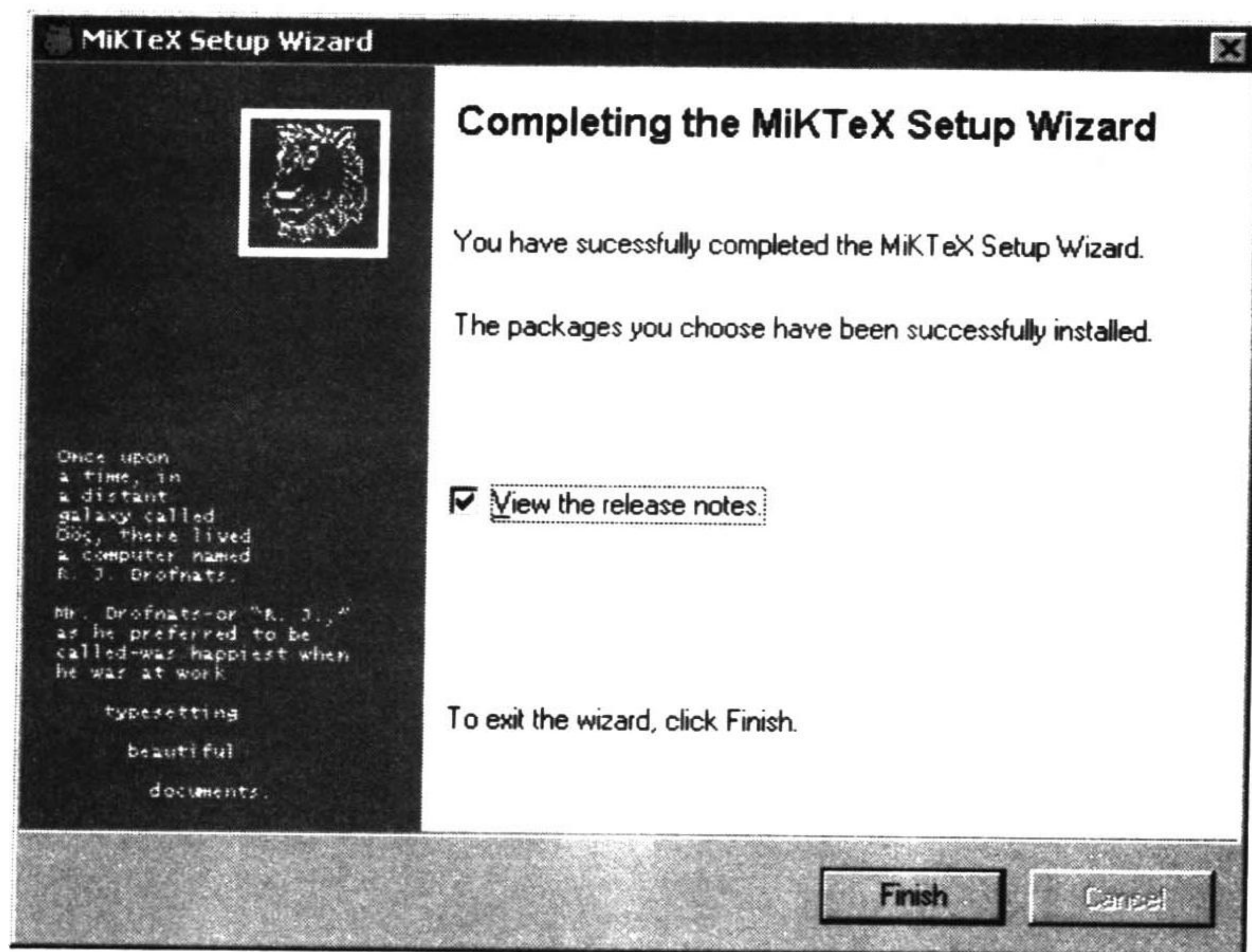
MiKTeX 建议你建立一个本机的 texmf 目录树, 其默认的路径是盘号加上 \Localtexmf. 在 MiKTeX 的运行过程中, \texmf 目录中的文件是不会改变的, 所有的更新文件都放在本机的 texmf 目录内. 点击‘下一步’后, 出现以下窗口:



你可采纳默认的选择(不要), 点击‘下一步’后, 出现以下窗口:



核对无误后, 点击‘下一步’, 会出现一个显示安装进度的窗口, 等安装完毕后, 出现以下窗口:



点击‘Finish’按钮, 安装就结束了, 为了使设置生效, 必须重新启动机器。

以后若需要添加 L^AT_EX 2_ε 的宏包, 可使用本附录 A1.1 节 (第 239 页) 介绍的方法。

A1.3.2 安装 WinEdt

安装 WinEdt 十分简单, 只要执行光盘上的 `\support\winedt\setup.exe`, 选择典型安装即可. 为了试验 WinEdt 是否正确运行, 可选取 `Project → Current Work (Samples) → AMS paper`, 当编辑窗口内出现文本后, 用鼠标点击工具栏上的 ‘LaTeX’ 图标, 就会弹出一个窗口进行编译, 并自动关闭. 如果仔细观察会发现窗口中有不少警告信息, 这是因为 paper 中含有文献引用, 因此接下去要执行 `Accessories → BibTeX`, 以后再执行两次 LaTeX, 就不再出现警告信息, 编译完成. 用鼠标点击工具栏上含有 ‘DVI’ 的图标 (光标放在上面时会出现 ‘DVI preview’ 的提示), 就会显示排版结果. 关闭预览窗口后点击工具栏上的 ‘dvi→pdf’, 不费吹灰之力就把 dvi 文件转换成 Acrobat 的 pdf 文件. 点击位于 ‘ps→pdf’ 按钮下面的 Acrobat Reader 的图标就能预览 pdf 文件 (当然你的机器必须安装了这个软件). 如果需要生成 ps 文件, 只需用鼠标点击工具栏上的 ‘dvi→ps’. 生成了 ps 文件后点击 ‘ps→pdf’ 按钮也能生成 pdf 文件.

需要使用 Postscript 图形功能的用户还需安装 Ghostscript 以及 GSView, 这两个软件的新版本都不是免费的, 不过未注册的用户仍可正常使用, 只是在每次启动时都会出现一个注册窗口. 要安装这两个软件只要执行光盘上 `\support` 目录内的 `gs700w32.exe` 以及 `gsview40w32.exe`. 这两个软件安装好后, 对于已经用 ‘dvi→ps’ 生成过 ps 的文件, 按一下工具栏上位于 ‘dvi→ps’ 按钮下面的 ‘GSView’ 钮就能预览 ps 文件.

A1.3.3 WinShell 的安装与配置

只要执行光盘上的 `\support\winshell120.exe` 就会自动安装 WinShell. 为了设置 WinShell, 我们选中 “Options → Program Calls”, 在弹出的窗口中利用 Browse 按钮为 DVIWin 与 Ghost View 选取它们所对应的执行程序, 例如: `yap.exe` 以及 `d:\ghostgum\gsview\gsview32.exe` (假定 GSView 安装在 D 盘). 用户利用 User defined 可以定义自己的工具, 例如可把 Tool 1 的 Name 栏取为 DviPdf, exe-File 栏则是 `dvipdfm.exe`, cmd 栏可输入参数 ‘`%pm.dvi`’, 并且去掉 LaTeX first 以及 DVIPS first 前面方框中打的勾, 点击 ‘确定’ 后退出. 再把用户自定义的工具程序放在工具条上, 方法如下: 选择 “Options → View → Customize”, 这时会弹出一个名为 ‘customize Toolbar / Macrobar’ 的窗口, 在左边 ‘Category’ 下方的下拉菜单里选取 ‘User-Programs’, 这时右边的 ‘Buttons’ 下面小窗口中显示了所有的工具程序, 用鼠标选中其中之一, 例如 ‘DviPdf’, 然后按住鼠标左键把这个程序拖放到工具条中, 在鼠标箭头成为一根竖直黑线的任何位置释放左键, 就会弹出一个名为 ‘Change

Button' 的新窗口, 这时左边显示 3 个无线电按钮: Image only; Text only; Image and Text. 这是让你选择工具条上图标的样式, 最简单的方法是选取 Text only, 点击 OK 后就在工具条上出现一个名为 'DviPdf' 的按钮, 如果选取 'Image only', 再选取 'Edit', 就会出现画有锤子图形的画板, 你可以自己设计一个图标. 以后用鼠标点击这个按钮就会执行名为 DviPdf 的工具程序. 这样打开一个 TeX 源文件后, 只要点击各个按钮就能完成所需的各项工作, 生成 dvi、ps 或 pdf 文件. 读者不妨用 small2e.tex 试试, 看看安装是否成功.

如果你使用本书光盘中所附的安装程序, 并且选中了 'WinShell 需用的天元补充文件', 那么工具栏里会出现 7 个已经定义好的按钮, 必要时你可用上面介绍的方法增加你的新按钮.

A1.3.4 天元的安装与运行

对于需要使用中文的读者, 可有 3 种选择: 单装天元, 单装 CJK, 或两者都装.

对于 WinEdt 的用户, 还需要在本书光盘的安装程序里选中 'WinEdt 需用的天元文件', 安装完成后, 工具栏里会增加 4 个已经定义好的按钮: 'AMSTeX', '天元', '天元 CFG' 以及 'FNDB'. 如果这 4 个新的按钮没有出现, 就请你再做一次安装, 不过只选 'WinEdt 需用的天元文件', 其它选项一概不选.

现在打开测试文件 \texuser\2-1-1.ty, 如果你的操作系统是 Windows2000 或 Windows ME, 则需要先点击 '天元 CFG' 按钮, 进入 Config → 字模文件名, 把其中的 simsun.ttf 全改为 simsun.ttc, 保存配置后退出. 然后点击 '天元' 按钮, 等天元处理完毕关闭窗口后, 再点击 'FNDB' 按钮, 完成后依次点击 'LaTeX' 按钮和 'DVI' 按钮, 就能显示排版的结果. 当天元能正常工作时, 一般不需要使用 '天元 CFG' 按钮. 这里需要说明的是 'FNDB' 按钮的意义, FNDB 是 File Name Database (文件名数据库) 的缩写, 在 MiKTeX 环境里的各种程序运行时需要读入的文件都要从这个数据库查找它的所在位置, 由于天元是外部软件, 它运行时新产生的 tfm 以及 pk 文件不会自动录入数据库, 如果不在运行 TeX 前手动运行 FNDB 的更新程序, TeX 运行时就会找不到所需汉字的 tfm 文件. 由此可见, 每当你在 \texmf 或 \localtexmf 目录里添加了新文件, 就应该更新 FNDB. 同理, 一个中文文件经天元处理后, 在修改过程中没有增加新的汉字, 就可以不更新 FNDB. 以后天元运行的次数多了, 就不必经常更新 FNDB 了. 当你用 TeX 编译一个文件, 屏幕上显示以下出错信息时:

```
!Font \ChineseFontzB=ccA14 not loadable: Metric <TFM> file not found.
```

就是提示你该用 FNDB 按钮更新数据库了.

类似地, 对于 WinShell 的用户, 本书光盘的安装程序已经替你安装了 7 个新

的按钮, 其中包括‘天元’、‘天元 CFG’以及‘FNDB’. 不过在 WinShell 环境中编译含有中文的 ty 文件时, 比较好的方法是选中 Options → Main-TeX-Document (工具条上有一个图标), 用 Browse 按钮选中与此 ty 文件对应的 tex 文件, 如果是第一次运行, 也可先选中 ty 文件, 然后再把扩展名改成 tex (必须如此改!). 这样即使你不打开源文件, 一按‘天元’钮就会自动执行天元程序处理指定的文件, 再按 LaTeX 又会对刚才生成的 tex 文件进行编译, 以后就与 WinEdt 一样处理.

如果你使用的打印机不是默认的 600DPI 的激光打印机, 那么还需要做以下修改. 进入预览程序 yap 后, 选取 View → Options → Printer, 选择正确的 Mode, 记下分辨率 (resolution), 再翻到 Display 页, 使得它与 Printer 有相同的 Mode. 点击‘天元 CFG’, 进入‘Config’, 把其中的密度参数修改得等于打印机的分辨率. 不过你若想生成 ps 或 pdf 文件, 那么还是应该把天元的密度设成 600DPI, 因为这是那些转换软件的默认分辨率, 也是比较合适的分辨率. 为此你只要在‘天元 CFG’的‘Config’中的‘重新生成 PK 文件’旁边打个勾 (同时去掉‘更新 Tab 文件’旁边的勾), 这时天元会以 600DPI 的密度重新生成一套字体, 供生成 ps 或 pdf 之用, 而又不影响预览或打印所需的字体.

实践证明, 即使用户的打印机不是 600DPI, 也可不作上述修改, 而总是把天元的密度设成 600DPI. 当生成 dvi 文件后, 把它转换成 ps 文件再打印, 效果是令人满意的.

A1.3.5 关于天元的配置

经验表明, 天元软件与 TeX 配合不好以至无法预览或打印汉字的原因都在于没有正确配置天元. 天元的基本思想是按字体的大小把天元源文件中出现的汉字编排成 128 个字的组, 例如 ccA10, ccB10, ... 内都是 10 pt 大小的汉字. 这些字库中哪个号码对应什么汉字都记录在文件 tycfnt.tab 中, 天元中被称为 Tab 文件. 根据 Tab 文件的记录, 天元把汉字‘翻译’成 TeX 能看懂的符号, 而且天元软件也会生成相应的尺寸 ccA10.tfm, ccB10.tfm, ... 以及点阵文件 ccA10.pk, ccB10.pk, ... 供 TeX 编译以及预览打印之用. 每次运行天元遇到新的汉字时, Tab 文件会添加记录, 当汉字太多时, 字体文件太多, 会影响天元的效率, 因此要清除 Tab 文件, 从零开始, 这就是天元 Config 的‘路径’页上‘更新 Tab 文件’的功能. 天元与 CJK 的不同, 就在于天元根据需要生成汉字点阵文件 (pk 文件), 而后者则是把 GB 或 GBK 编码集的所有汉字分组, 每个汉字的分组是固定的, 一个组中有一个汉字被用到, 就要生成整个组的汉字点阵, 而且不同尺寸及不同字体要生成不同的文件, 所以 CJK 所需的硬盘空间远大于天元.

进入天元的 Config 后, ‘路径’页上的信息是最重要的, 如果你使用的是 MiK-

TeX, 而且安装在 D 盘上, 那么‘TFM 文件路径’应该是 `d:\localt~1\fonts\tfm\`, ‘PK 文件路径’应该是 `d:\localt~1\fonts\pk\dpi@Rr\`, 假如你使用 ttf 字模并且 Windows 的路径是 `c:\windows`, 那么‘字模文件路径’应为 `c:\windows\fonts\`. 这 3 个路径如果填得与实际情况不符, 在 TeX 编译时会报错, 或在预览打印时没有汉字出来. 还有‘密度’栏应该填写你的打印机的分辨率, HP 打印机总是 300 或 600DPI, 而 Canon 系列一般是 360 或 720DPI, 一定要搞清楚, 填入正确的值, 否则就不能打印中文. 如果你的打印机的型号能在预览器 `yap.exe` 的列表中找到, 那么你就可以设置成它上面指示的分辨率. 另一栏‘Magnification’中填写的数字应该与天元源文件第一行 `\def\ChineseScale` 命令后面的括号里的数字一致. 一般说来, 上面这些信息填写得正确就不会出现无法显示或打印汉字的问题.

在天元 Config 的‘路径’页上还有一个‘重新生成 PK 文件’的选择, 它的含义是按照 Tab 文件上的记录生成 pk 文件, 主要用于改变密度的情形, 例如你为了生成 pdf 或 ps 文件, 应该选用默认密度 600DPI, 而你的打印机又是 360DPI 的, 那么你只需在 600DPI 下完成一切工作后再把天元的密度设成 360DPI 并选中‘重新生成 PK 文件’, 在你退出 Config 后天元会自动生成所需的 pk 文件, 你不用再编译就能直接打印刚才的文件. 因此, 相邻的两个选项‘重新生成 PK 文件’以及‘更新 Tab 文件’不应同时选中, 否则会使天元陷入混乱.

A1.3.6 安装 CJK

首先用户应该安装与 CJK 相关的文件, 为此选中

开始 → 程序 → MiKTeX → MiKTeX Options

出现相应窗口后选取‘Packages’页, 先在‘Download Site’栏里选中你的光盘内的目录 `\miktex\tm\packages`, 再在左边小窗口的‘Languages’左边的田上点击, 就会显示下一层目录, 用鼠标点击‘Chinese, Japanese, Korean’左边的方框, 去掉方框内的阴影, 表示选中. 然后点击‘应用’, 就开始自动安装 CJK 文件包. 为了检验 CJK 是否安装成功, 可以把 `d:\texuser\2-1-1cjk.tex` 作为测试文件.

在本书光盘的安装程序里已经替你设置了以下几种字体: 宋、楷、仿宋、黑、隶书与圆体, 它们都要调用你的 Windows 自带的 ttf 字库, 一般前面 4 种总是有的, 而后面两种则不一定, 请先试了再用.

最后介绍如何在 CJK 中添加 ttf 字体, 目前无此需求的读者可以跳过这部分内容. 假定你有一个华文新魏字体, 其文件名是 `stxinwei.ttf`. 如果这个文件既不在你的 Windows 目录内的 `\fonts` 下面, 也不在 `\localtexmf\fonts\truetype` 下面, 那么首先要修改以下配置文件:

`\localtexmf\miktex\config\miktex.ini`

在里面找到以下内容:

```
TTFPath=.;%R\fonts\truetype//;%windir%\fonts//
```

并在行末添加分号;后加上ttf文件所在的路径,再以//结尾.然后打开文件

```
\localtexmf\ttf2tfm\base\ttf fonts.map
```

按以下两行的形式

```
gbkfs@UGBK@ simfang.ttf Pid = 3 Eid = 1
```

```
gbkfssl@UGBK@ simfang.ttf Slant=0.25 Pid = 3 Eid = 1
```

添加两行(如果是GB编码,则要把UGBK改为UGB):

```
gbkwei@UGBK@ stxinwei.ttf Pid = 3 Eid = 1
```

```
gbkweisl@UGBK@ stxinwei.ttf Slant=0.25 Pid = 3 Eid = 1
```

修改完毕后进入\localtexmf\fonts\tfm\chinese\gbkwei目录,执行以下两条命令:

```
ttf2tfm stxinwei gbkwei@UGBK@
```

```
ttf2tfm stxinwei -s 0.25 gbkweisl@UGBK@
```

就能生成相应的直立体与斜体字的tfm文件.再把文件\localtexmf\tex\latex\cjk\gb\c19fs.fd复制成\localtexmf\tex\latex\cjk\gb\c19wei.fd,并把文件中的fs都替换成wei.把FNDB更新后(例如使用命令initexmf -u),一切都完成了.以后的字体wei就是华文新魏字体.

A1.4 如何在DOS环境下安装和使用emTeX

对于硬件配置较低的用户,可以安装体积较小的emTeX,直接在DOS环境下使用TeX.为了方便用户,我们提供了一个批处理程序文件,但首先要安装emTeX和天元软件,有经验的用户可自行按照软件本身附带的说明文件进行安装,并修改批处理文件中设置的路径.对于新手或者想省事的用户,可简单地如下操作:

1. 直接将光盘根目录下的子目录\emt看整个复制到硬盘任意盘符(例如D盘)的根目录下.
2. 将光盘\emt看目录中的文件tytex.bat复制到环境变量PATH包含的任一路径中,例如复制到C:\DOS或C:\Windows\command中.
3. 修改文件\emt看\ty\pkttf.cfg,将其中的C:\Windows改为实际的目录,例如你的Windows目录是C:\PWIN98,就将文中的相应内容改为C:\PWIN98.

如果计算机上没有安装Windows 9X,可省略这一步;或者按照天元说明文件自行安装ttf中文字库,并将文件\emt看\ty\pkttf.cfg中-F选项的值改成ttf字库的路径.

RootName = 2-1-1

TyTeX.bat (Math. ECNU 1997.1)	
1 = Config TyTeX	6 = LaTeX
2 = Ty (ttf)	7 = AmsTeX
3 = Ty (sl)	8 = TeX
4 = Reset	9 = View, Print
5 = Edit	0 = Other
0 = DOS, Help-View, My1.bat, ...	
X = Quit	

Strike a key (0-9,X): _

图 A1.4-1 TyTeX.bat 主菜单

通常还要建立一个目录作为用户的工作目录, 例如用 `\texuser` 作为工作目录. 可将光盘 `\examples` 目录中的文件复制到工作目录以进行下面的练习.

为了简化批处理文件, 要求含有中文的源文件用扩展名 `ty`, 纯西文源文件用扩展名 `tex`. 运行批处理程序的格式为 `tytex 文件名`, 不写扩展名, 程序会自动查找 文件名 `.ty` 或 文件名 `.tex` 文件, 前者优先. 若这两个文件都不存在, 则会询问是否建立相应文件.

如果当前目录有文件 `2-1-1.ty`, 运行 `tytex 2-1-1` 后屏幕将显示主菜单如图 A1.4-1.

按 `X` 键可退出批处理程序. 主菜单中其他选项的功能如下:

1 - Config TyTeX: 修改天元的配置文件 `tyc.cfg`. 选中这一项后将显示配置文件的内容供用户修改, 文件中每行是由 “-” 引导的单字母选项及其所带参数, 中间没有空白, 选项字母的大小写是有区别的. 部分选项如下:

-d 项的值可用 180、300、600 等, 分别对应打印机密度 180、300、600 点. 大多数针式打印机是 180 点, 低价位喷墨和激光打印机是 300 点, 中档的是 600 点. 其他密度也可使用, 例如对于 1016 点的照排机, 使用 `-d1016`. 默认安装的取值是 `-d600`.

-m 项用于控制中文的放大倍数, 其值必须与源文件中放在导言区的命令 `\def\ChineseScale{...}` 中的数值相同. 默认安装的取值是 `-m1000`.

-o 项的值是输出的 `tex` 文件路径, `.\` 表示当前目录.

-T 项的值是 `tycfnt.tab` 文件的路径. 若无该项, 则默认的路径是 `tyc.exe` 所在的子目录.

-K 项的值是字宽 (单位是 mm), 默认值是 3.24629, 使用 10 pt 的 L^AT_EX 时也可选用 3.47133.

-S 项的值是字深(单位是mm), 默认值是0.61834, 使用10 pt的 \LaTeX 时也可选用0.71967.

-J 项的值是字间距(两个相邻汉字间的距离, 单位是mm), 默认值是0.23187, 使用10 pt的 \LaTeX 时也可选用0.21167.

-O 项的值是阔体字的字宽与整个字高的比, 默认值是1.2.

-U 项的值是瘦体字的字宽与整个字高的比, 默认值是0.83333.

平常使用时, 仅偶尔需要修改-m的值, 其它值一般是不需要改动的. -P和-M的值已由本批处理程序设定, -F的值已不再被`tyc.cfg`使用, 所以这3个选项不必出现在`tyc.cfg`中.

2 - Ty (ttf): 运行天元程序, 产生扩展名为`tex`的文件. 该项按照文件`\emtex\ty\pkttf.cfg`中-F选项的值查找ttf字库的路径, 使用TrueType Font中文字库获取所需要的汉字.

3 - Ty (sl): 运行天元程序, 产生扩展名为`tex`的文件. 该项按照文件`\emtex\ty\pksl.cfg`中-F选项的值查找矢量中文字库获取所需要的汉字. 与前一条命令一样, 必须有这种字库才能使用这个命令. 如果在计算机上安装了UCDOS 3.0的矢量中文字库, 路径是`e:\ucdos\fonts`, 就要把文件`\emtex\ty\pksl.cfg`中的-F项改为`-Fe:\ucdos\fonts`. 因涉及版权问题, 光盘中没有提供上述任何一种中文字库.

4 - Reset: 运行天元的重置命令. 当多次运行天元后, 如果出现显示错误, 就需要使用一次这个命令.

5 - Edit: 编写或修改源文件. 在编辑状态, 按F1键显示帮助信息(再按F1键显示更多帮助信息), 按F3键后再按E键就是存盘退出, 按F3键后再相继按Q键和Y键则是放弃修改并退出编辑状态.

6、7、8 - 根据源文件的不同格式, 选用不同的 \TeX 编译命令. 本书中的例子需用 \LaTeX 编译, 故应选6.

9 - View, Print: 在屏幕上观看或在打印机上打印DVI文件(排版结果). 使用这一项时会显示一个菜单, 见图A1.4-2. 应根据主菜单第1项修改天元配置文件时给出的-d值, 选用对应的View或Print命令. 菜单上方给出了常用Options(选项)的例子, 不输入选项时可直接回车. 详细的选项值及其含义请参阅 \TeX 系统所带的说明文件, 此处例子仅作备忘用. 在View状态, 可按+、-号放大或缩小显示内容, 按Q键退出显示状态.

0 - 其他一些命令, 使用这一项时也会显示一个菜单, 见图A1.4-3.

在图A1.4-3中, 第1项的用途是在不退出批处理程序的情况下使用DOS命令, 可用`exit`命令返回批处理程序.

```

===== Option examples =====
I Options=@2onl-A4 Options=/tr3      I
I Options=/h197mm Options=/w297mm I
I Options=/b10      Options=/e30      I
I Options=/do       Options=/de       I
I Options=/z                I
=====
***** View ***** Print *****
*      1 - View (180)      2 - Print (180)      *
*      3 - View (300)      4 - print (300)      *
*      5 - View (600)      6 - Print (600)      *
*****
*                                *
*              0 - Return              *
*****
Strike a key (0-6): _

```

图 A1.4-2 TyTeX.bat 的 View, Print 菜单

```

***** Other *****
*      1 - DOS      *
*      2 - Help (view)  *
*      3 - my1.bat   *
*      4 - my2.bat   *
*      5 - my3.bat   *
*      6 - my4.bat   *
*      7 - my5.bat   *
*      _____   *
*      0 - Return    *
*****
Strike a key (0-7): _

```

图 A1.4-3 TyTeX.bat 的其他选项

第2项是显示DVI文件浏览器的帮助信息。

第3、4、5、6、7项是调用用户自编的批处理文件, 文件名是my1.bat, my2.bat, my3.bat, my4.bat 和 my5.bat, 放在\emt看\ty目录, 用户应根据实际需要自行编写这些文件。

当源文件为西文tex时(不含任何中文命令以及Tydraw作图命令), 可同样使

用 `tytex.bat` 批处理程序, 只是不再使用主菜单中与中文处理有关的前 4 条命令。

A1.5 如何在 PCTeX32 中使用天元

PCTeX32 是 Windows 操作系统下的一个 TeX 集成环境, 它是商业软件, 但功能不及 MiKTeX+WinEdt. 考虑到仍有部分人还在使用 PCTeX32, 下面介绍在这个环境中使用天元软件的方法。

为了便于说明问题, 假设已将 PCTeX32(v4) 安装在 `E:\pctexv4` 目录. 设用户工作目录为 `D:\texuser`, 设 Windows 目录为 `C:\windows`, 已将 `pctexv4.ini` 复制到 `C:\windows` 目录, 并已根据实际目录结构修改了该文件中的路径. 为了在 PCTeX32 中使用天元, 应先做下列事情:

1. 建立如下目录

`e:\pctexv4\cc\ty` (存放天元 `tywin` 的系统文件)

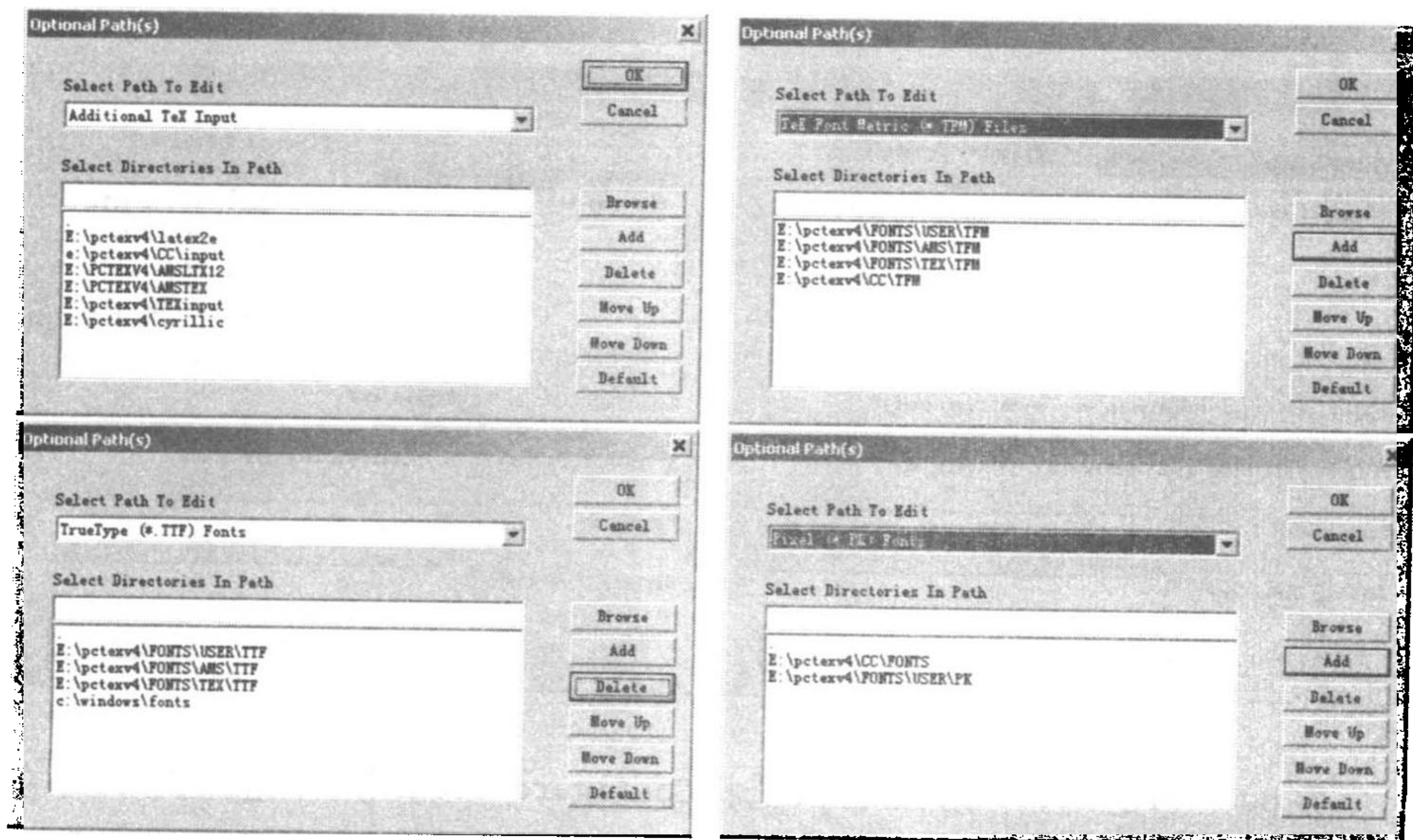
`e:\pctexv4\cc\input` (存放自行设计的格式文件及 `tdw.tex` 等)

`e:\pctexv4\cc\tfm` (存放固定的或动态生成的汉字 `tfm` 文件)

`e:\pctexv4\cc\fonts` (含子目录 `dpi600`, `dpi657` 等, 存放 `*.pk` 文件)

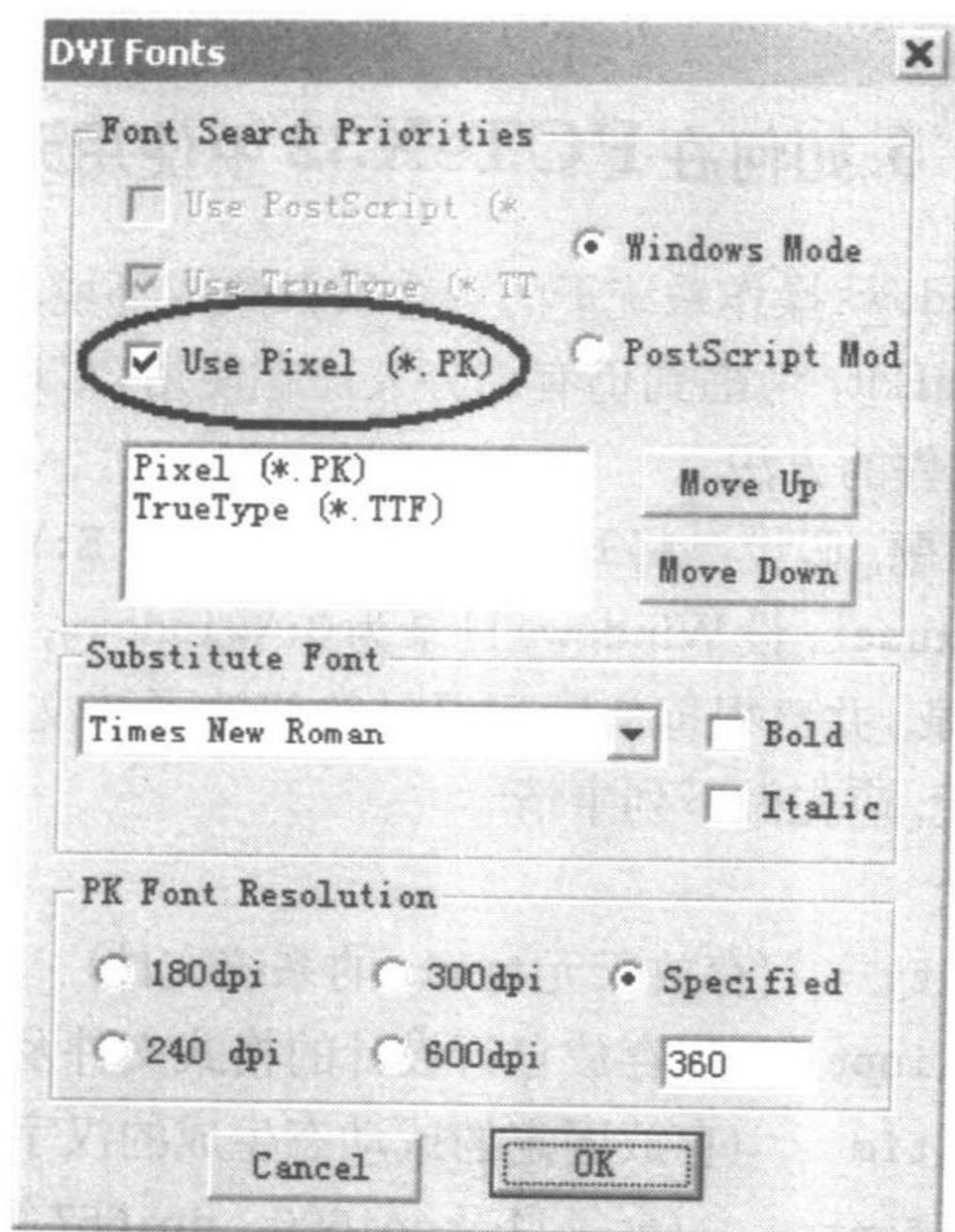
2. 添加路径

运行 PCTeXv4, 打开 **Settings** 菜单, 选择 **Directories...** 项, 按照下列几图添加路径. 其中与 `cc` 目录有关的路径都是新加上去的。



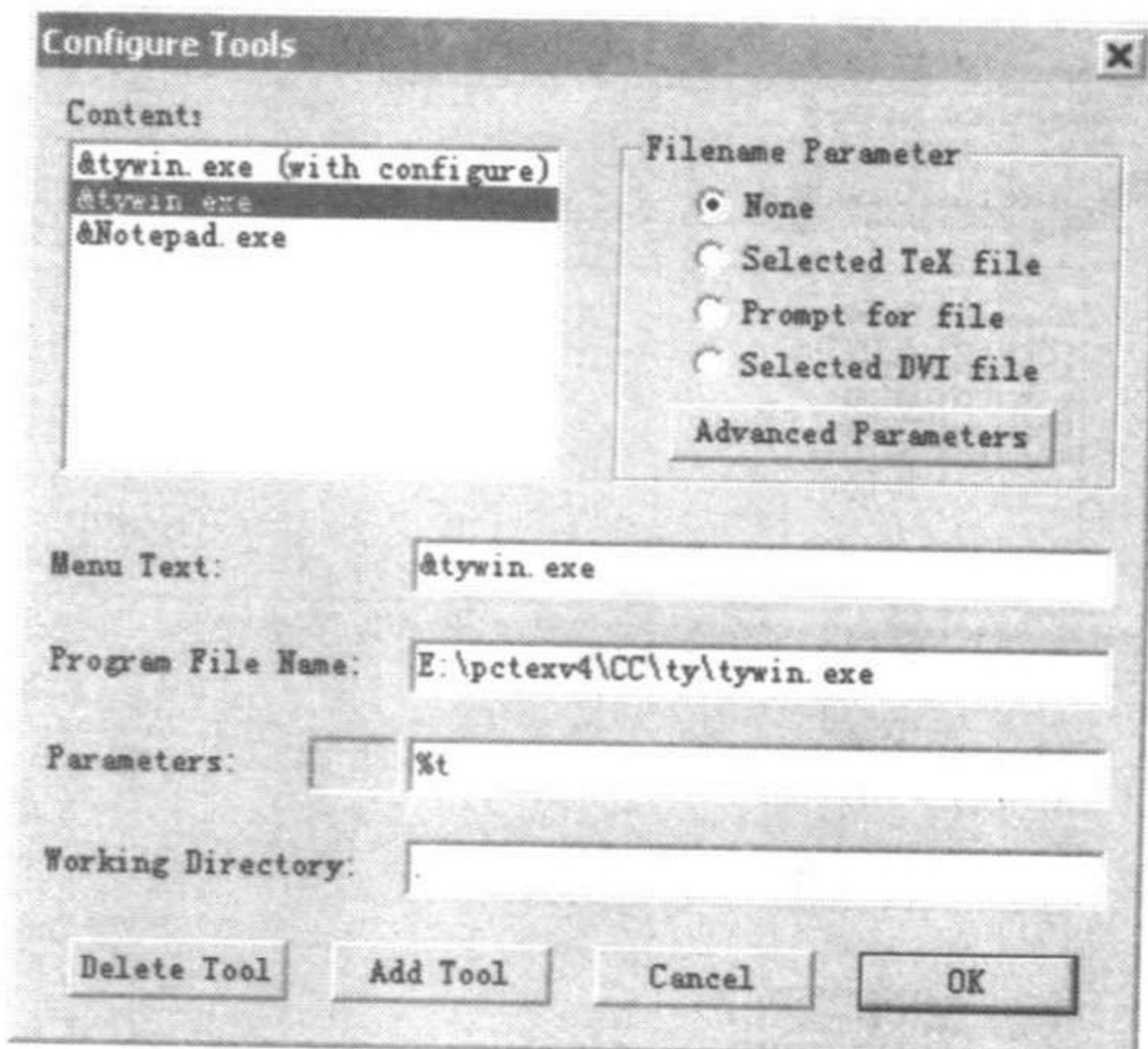
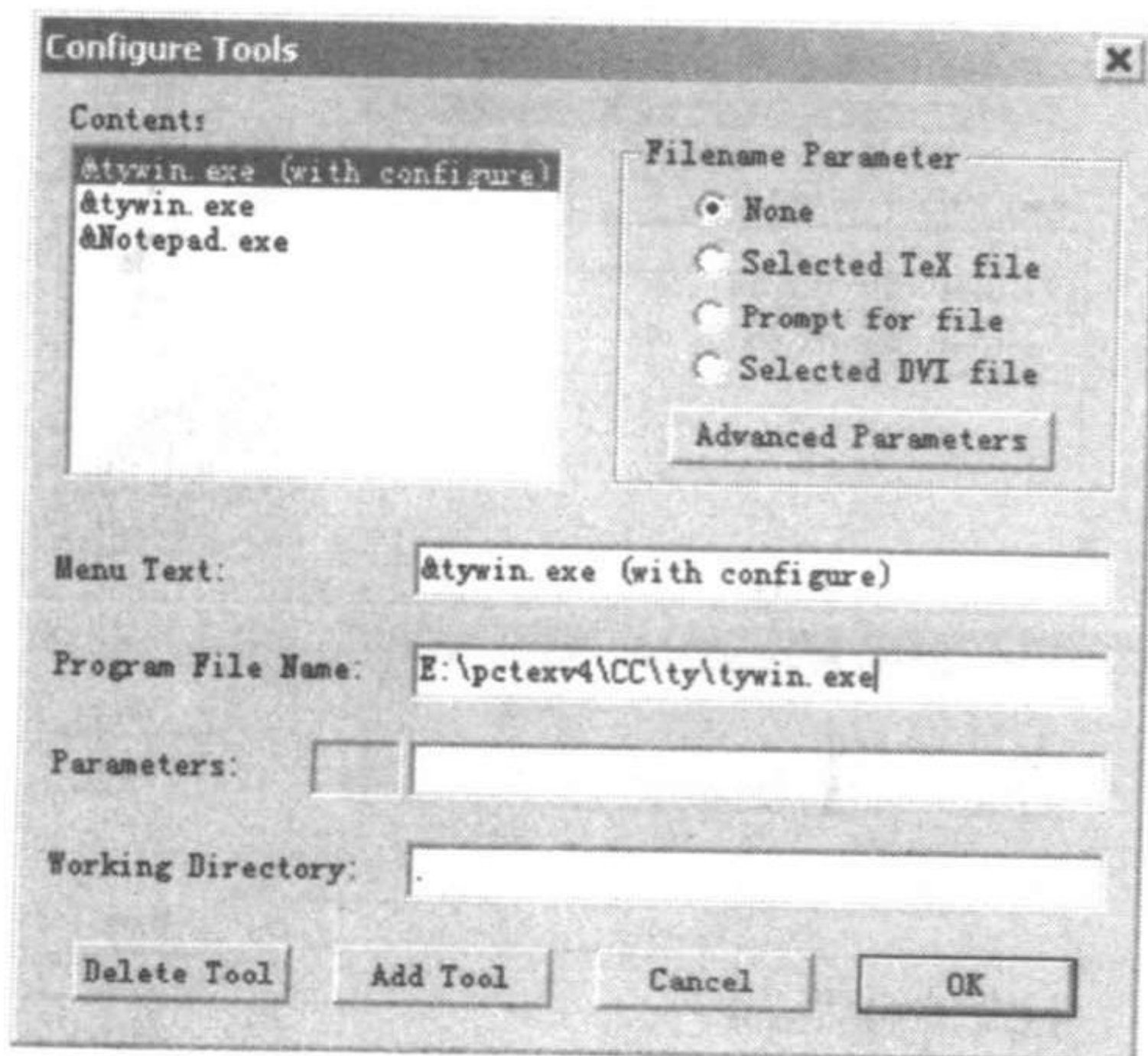
3. 在 PCTeXv4 的 **Settings** 菜单, 选择 **DVI Fonts...** 项, 一定要勾选图中所

圈的 Use Pixel (*.PK) 项:



此外, 选择的 PK Font Resolution 必须与天元设定的密度相同 (例如都是 600DPI).

4. 建立天元工具



在 pctexv4 的 Settings 菜单中, 选择 Configure Tools 项, 单击 Add Tool, 在随即出现的对话框中选择应用软件后就会自动填写几个栏目, 可修改 Menu Text 栏的内容, 改为自己喜欢的名字. 通常还要修改工作目录, 当前目录用 “.” 表示,

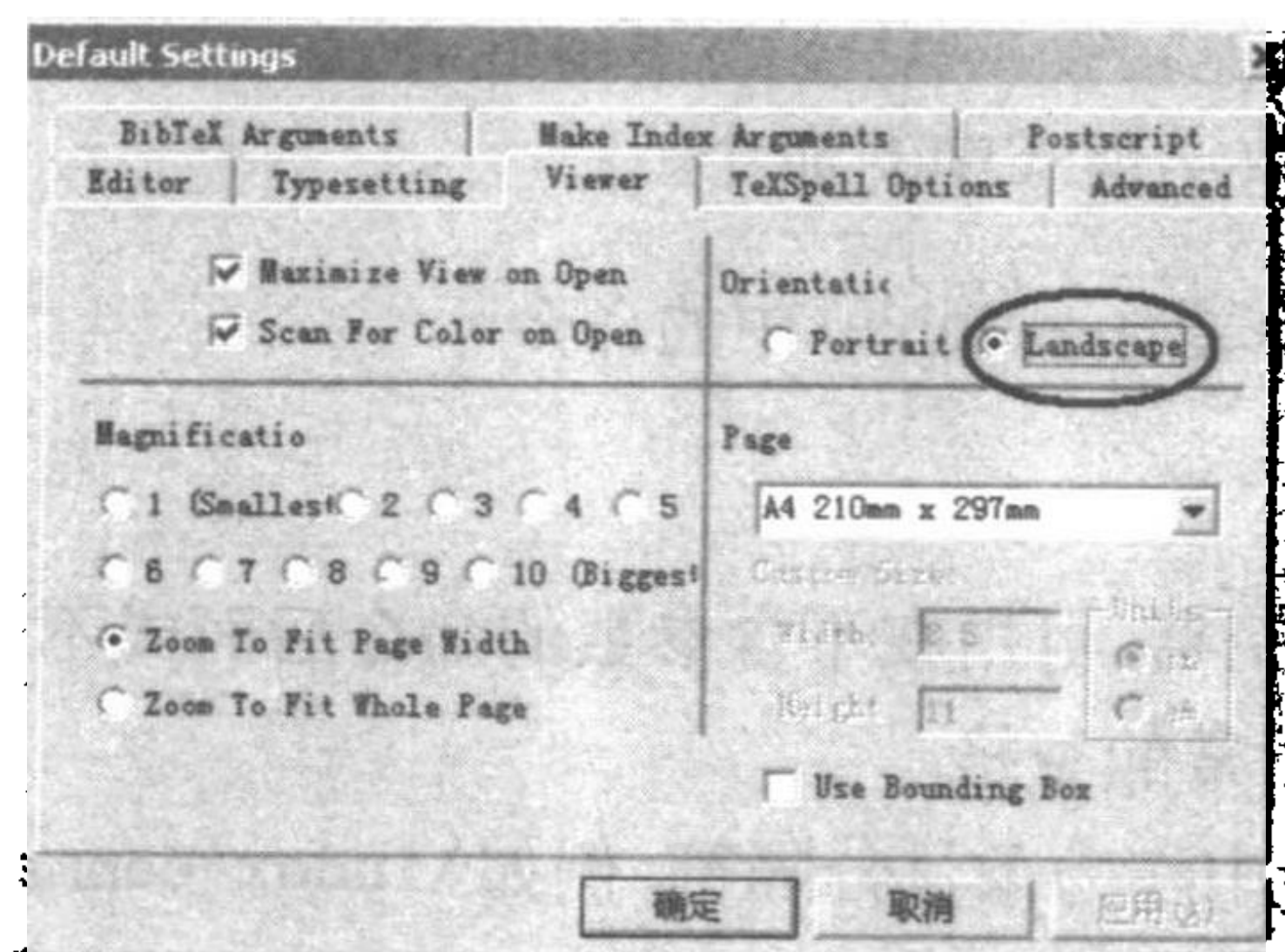
也可改为固定的工作目录, 例如改为 `D:\texuser`. 当工作目录是 “.” 时, 在第一次使用工具时选择的目录就是以后使用该工具时自动出现的目录.

第一个工具是 `tywin.exe (with config)`, 由于未指定命令行参数, 所以出现配置对话框, 此时可用 **Config** 按钮对天元进行配置, 或用 **Open** 按钮打开天元源文件.

第二个工具也是 `tywin.exe`, 但指定了命令行参数, 所以不出现配置窗口, 这对不需要经常变动配置的大多数用户是更适用的选择. 根据有无命令行参数而决定是否出现配置窗口, 这是天元作者为方便用户所设置的功能.

其他工具可类似设置.

附注: 若在 `LaTeX` 文件中使用了 `landscape` 选项, 则必须在 **Viewer** 选项卡中使用同样的选项, 具体设置见下图 (图中所圈之处):



附录二 L^AT_EX 命令简介

本附录包含了所有 L^AT_EX 命令的简介, 命令均按 ASCII 字母次序排列, 但出现在第一位的 \ 不参加排序.

有的命令后面带有记号, 其意义如下:

[a] 是属于 *AMS-L^AT_EX* 的命令, 为使用此类命令, 须在前言部分输入宏包 `amsmath`;

[m] 是只允许出现在数学模式中的命令;

[p] 是只允许出现在导言 (preamble) 中的命令.

_ 产生一个正常的空格, 可用在不含参数的命令后面或不表示句子结束的句点后, 以产生一个正常的空格.

! 用在 `\index` 命令中作为条目分隔符, 例如 `\index{command!fragile}` 可产生一个索引, 其主条目是 'command', 副条目是 'fragile'.

!‘ 产生 ¡.

\! [m] 在数学模式中产生一个长度为 $-1/6$ quad 的空格 (即后退这段距离): 例如 `xx\!x = xxx`.

" 1. 在正常的文本模式中产生一个右双引号”;

2. 用于 MakeIndex 中时, 表示按字面意义打印紧跟在后面的特殊字符 !, @, | 或 ", 例如: `\index{"!}` 表示产生字符 ! 而不把它看成条目分隔符;

3. BibT_EX 中文本域的分界符, 例如: `AUTHOR = "Donald E. Knuth"`.

\" 产生重音号 umlaut: `\"{a} = ä`.

在用户自定义命令或环境时被用作变量替换符.

在嵌套的用户自定义命令或环境时被用作内部变量的替换符.

\# 产生字符 #.

\$ 用于在文本模式与行内数学模式之间进行切换的开关字符, 它的首次出现 (从文本到数学) 相当于 `\(` (或 `\begin{math}`), 它的再度出现 (从数学到文本) 相当于 `\)` 或 `\end{math}`.

- $\backslash \$$ 产生美元号 \$.
- % 注解符号, 从这个字符开始直至行末的字符均被 T_EX 忽略.
- $\backslash \%$ 产生百分号 %.
- & 在 `array` 以及 `tabular` 环境中用来表示开始一个新的列.
- $\backslash \&$ 产生字符 &.
- $\backslash '$ 1. 产生一个重音号: $\backslash 'a = \acute{a}$;
2. 在 `tabbing` 环境中, 表示跳到当前列的末尾, 使此命令左边的文本右对齐.
- () 在 `picture` 环境的绘图命令中用于指定用一对数表示的坐标; 在 BibT_EX 中可用于条目类型的最外层分组的符号 (通常用的符号是 { }).
- $\backslash ($ 从文本模式转换成行内数学模式的开关符, 其作用相当于 `\begin{math}` 或文本模式下的 \$ 号.
- $\backslash)$ 从行内数学模式转换回文本模式的开关符, 其作用相当于 `\end{math}` 或数学模式下的 \$ 号.
- $\backslash +$ 在 `tabbing` 环境中表示从下一行开始左边的边界移到下一制表位 (跳过一列), 这个命令只能出现在每行的头或尾.
- $\backslash ,$ 产生一个长度等于 1/6 quad 的小间隔, 可用于文本或数学模式, 例如: `xx\,x = xx x`.
- 产生连字符 -; -- 产生数字间的连接号 (en dash) -; --- 产生西文破折号 (em dash) —.
- $\backslash -$ 1. 表示隐藏的连字符, 如果在一个西文单词中间含有 $\backslash -$, 那么这个词的正常断字换行规则将失去作用, 当需要断字换行时, 只能选在出现命令 $\backslash -$ 的位置, 并插入连字符;
2. 在 `tabbing` 环境中表示从下一行开始左边的边界移到前一个制表位, 也就是抵消一个 $\backslash +$ 的效果.
- $\backslash .$ 产生一个重音号: $\backslash .o = \acute{o}$.
- $\backslash /$ 当斜体转为正体 (直立体) 时, 用此命令产生一个额外留空, 此外, 此命令也可用于避免某些西文字母间的连字 (ligature), 如 `f\ /f = ff`, 比较 `ff = ff`.
- $\backslash :$ [m] 在数学模式中产生一个中等长度 (2/9 quad) 的留空, 如: `xx\ :x = xx x`.
- $\backslash ;$ [m] 在数学模式中产生一个较大 (5/18 quad) 的留空, 如: `xx\ ;x = xx x`.
- $\backslash <$ 在 `tabbing` 环境中出现在行的首部, 表示当前行的左边界往左移动一个制表位, 在此之前应该已用 $\backslash +$ 命令使左边界往右移动了若干个制表位, 否则会出错.
- $\backslash =$ 1. 产生一个重音号: $\backslash =o = \bar{o}$;
2. 在 `tabbing` 环境中表示在当前位置重设或添加 (不是插入) 一个制表位.

- `\>` 在 `tabbing` 环境中表示往右进到下一个制表位.
- `?'` 产生 $\dot{}$.
- `@` 1. 在 `MakeIndex` 中, 用于把 `\index` 命令的参数分成两部分: `@` 前的部分是供条目按字母排序用的, 后面的部分则是打印输出的内容, 例如:
`\index{sum@$ \sum$}` 表示该条目按 `sum` 排序, 而打印出来的是求和号 Σ ;
 2. 在 `BibTEX` 中表示条目类型, 例如 `@BOOK` 表示以下的条目都是 `BOOK` 类型的.
- `\@` 插在句子末尾的大写字母与句点之间, 说明这个句点确实是句子的结束, 而不是跟在大写字母后面表示缩写词的, 提示 `TEX` 在句点后面多留一点空.
- `[]` 为命令或环境提供可选的参数.
- `\[` 从文本模式转换成行间数学模式的开关符, 使后面的数学公式单独成一行, 其作用相当于 `\begin{displaymath}`.
- `\\[长度]` 立即换行(不保持右端对齐), 其可选的参数长度表示在下一行之前插入一段竖直方向的空白.
- `*[长度]` 作用类似于 `\\`, 不过禁止在当前行与下一行之间分页.
- `\]` 从行间数学模式转回文本模式的开关符, 其作用相当于 `\end{displaymath}`.
- `^` `[m]` 数学式中的指数或上标, 例如: $x^2 = x^2$, $x^{-2n} = x^{-2n}$.
- `\^` 产生重音号: `\^o = ô`.
- `_` `[m]` 数学式中的下标, 例如: $a_n = a_n$, $a_{i,j,k} = a_{i,j,k}$.
- `_` 产生下划线: `t_v = t.v`.
- `\'` 1. 产生重音号: `\'o = ò`;
 2. 在 `tabbing` 环境内, 使它后面的文本右对齐到右边界, 因此在这个命令以后不能再出现 `\>` 或 `\=`.
- `{ }` 1. 在调用一个命令或环境时, 用来提供指定的变量;
 2. 对文本分组, 创建一个不命名的环境;
 3. 在 `BibTEX` 中作为条目类型的文本的分隔符, 也可作为文本域的分隔符.
- `\{` 产生左花括号 `{`.
- `|` `[m]` 产生 `|`.
- `|` 在 `MakeIndex` 中, 是 `\index` 命令内的命令字符, 相当于正常情况下的字符 `\`.
1. 在定义 `\newcommand{\ii}[1]{\textit{\textit{#1}}}` 后, 命令 `\index{entry|ii}` 的效果是在索引中把条目 'entry' 的页码用 `\textit` 字体打印;
2. 在 `makeidx.sty` 中定义的交叉引用命令 `\see` 可以在 `\index` 内调用, 如命令 `\index{bison|see{buffalo}}` 产生在索引内的交叉引用.
- `\|` `[m]` 产生 `||`.

`\}` 产生右花括号 `}`.

`~` 产生单词间的正常空格, 但是禁止在这里换行, 例如 `Prof.~Jones` 保证 ‘Prof.’ 和 ‘Jones’ 位于同一行.

`\~` 产生重音号 tilde: `\~n = ñ`.

`\a=` 在 `tabbing` 环境内产生原本由 `\=` 定义的重音号: `\a=o = ō`.

`\a'` 在 `tabbing` 环境内产生原本由 `\'` 定义的重音号: `\a'o = ó`.

`\a‘` 在 `tabbing` 环境内产生原本由 `\‘` 定义的重音号: `\a‘o = ò`.

`\AA` 产生 Å.

`\aa` 产生 å.

`\abovedisplayskip [m]` 一个长的行间单列公式与它前面的文本行之间的竖直间隔, 可用 `\setlength` 命令设置新值:

`\setlength{\abovedisplayskip}{10pt plus2pt minus5pt}`

`\abovedisplayshortskip [m]` 一个短的行间单列公式与它前面的文本行之间的竖直间隔, 同上例, 可用 `\setlength` 命令设置新值.

`\abstractname` 含有摘要标题的命令, 在英语中定义为 ‘Abstract’, 但可被重新定义以适用于其他语言.

`\acute{x} [m]` 用于数学变量 x 上的重音号: `\acute{a} = á`.

`\addcontentsline{文件类型名}{格式}{条目}` 用手工方式把条目加到文件类型名所规定的目录文件 `toc`, `lof` 或 `lot`, 参数格式是指采用哪一种章节命令的标题格式, 例如:

`\addcontentsline{toc}{section}{References}`

`\address{发信人地址}` 用在文件类 “信件” (`letter`) 中, 以输入发信人地址, 如不止一行, 可用 `\\` 换行.

`\addtime{秒}` 在文件类 “投影片” (`slide`) 中, 如果选取了选项 `clock`, 在 ‘注记’ (`note`) 的底部会出现时间标记, 时间标记以分为单位, 可用命令 `\settime` 设定, 本命令把指定的秒数加到时间标记中.

`\addtocontents{文件类型名}{条目}` 用手工方式把条目加到文件类型名所规定的目录文件 `toc`, `lof` 或 `lot`, 例如:

`\addtocontents{lof}{\protect\newpage}`

`\addtocounter{计数器}{数}` 把一个数加到计数器中存储的当前值上.

`\addtolength{\长度名}{长度}` 把一个长度加到长度命令 `\长度名` 的当前值上.

`\addvspace{长度}` 在本命令所在的段落后插入指定长度的竖直间隔, 此间隔被叠加到已有的间隔上, 但总和不超过长度所指定的值.

`\AE` 产生 Æ.

`\ae` 产生 æ .

`\aleph [m]` 产生 \aleph .

`\allowdisplaybreaks[数] [p][a]` 本命令允许在多行的数学公式中间换页, 可选项 [数] 的值可取 0-4, 其值越大, 换页越容易发生, 如果不使用这条命令, 也可以手工换页, 只要在公式的末尾加上命令 `\displaybreak`.

`\Alph{计数器}` 用大写字母打印计数器的当前值.

`\alpha [m]` 产生 α .

`\alsoname` 用于修改宏包 `makeidx` 的命令, 在英文中, 命令 `\seealso` 的文本是 'see also', 可以重定义此命令以适合其他语言.

`\amalg [m]` 产生 \amalg .

`\and` 用在 `\author` 命令中区分作者名, 供 `\maketitle` 生成标题页.

`\angle [m]` 产生 \angle .

`\appendixname` 含有附录标题的命令, 在英语中定义为 'Appendix', 但可被重新定义以适用于其他语言.

`\approx [m]` 产生 \approx .

`\arabic{计数器}` 用阿拉伯数字打印计数器的当前值.

`\arccos [m]` 产生数学公式内的函数名 'arccos'.

`\arcsin [m]` 产生数学公式内的函数名 'arcsin'.

`\arctan [m]` 产生数学公式内的函数名 'arctan'.

`\arg [m]` 产生数学公式内的函数名 'arg'.

`\arraycolsep` 在 `array` 环境中列与列之间留空宽度的一半, 可用 `\setlength` 命令为其指定新的值:

```
\setlength{\arraycolsep}{3mm}
```

`\arrayrulewidth` 在 `array` 以及 `tabular` 环境中竖直或水平线的厚度, 可用命令 `\setlength` 为其指定新的长度:

```
\setlength{\arrayrulewidth}{0.5mm}
```

`\arraystretch` 用来改变表格内的行间距的倍数, 其正常值等于 1, 实际间距等于已定义的行间距乘以此倍数, 可用以下命令为其指定新值:

```
\renewcommand{\arraystretch}{倍数}
```

`\ast [m]` 产生 $*$.

`\asymp [m]` 产生 \asymp .

`\AtBeginDocument{命令序列} [p]` 先把命令序列存储起来, 等到执行命令 `\begin{document}` 时再把这段命令序列插入待处理的数据流中, 命令序列的内容可以包括只允许出现在导言中的命令, 在制作宏包时可利用这条命令

以保证某些命令不会被别的宏包所覆盖.

`\AtEndDocument{命令序列}` [p] 先把命令序列存储起来, 等执行`\end{document}`时再把这段命令序列插入待处理的数据流中, 在制作宏包时可利用这条命令在结束文件时自动打印一些附加的内容.

`\AtEndOfClass{命令序列}` [p] 先把命令序列存储起来, 等到当前类文件输入完毕时再把这段命令序列插入待处理的数据流中. 本命令只能出现在类文件或被类文件读入的文件中, 常被用于用户的本地配置文件中, 它可能在类文件的起始部分被读入, 而此本地配置又要在类文件结束时覆盖默认的配置.

`\AtEndOfPackage{命令序列}` [p] 先把命令序列存储起来, 等到当前宏包文件输入完毕时再把这段命令序列插入待处理的数据流中. 本命令只能出现在宏包文件或被宏包文件读入的文件中, 常被用于用户的本地配置文件中, 它可能在宏包文件的起始部分被读入, 而此本地配置又要在宏包文件结束时覆盖默认的配置.

`\author{名字}` 输入作者名以供`\maketitle`命令生成标题页.

`\b{x}` 产生位于字母下的重音号: `\b{o} = o`.

`\backmatter` 在文件类“书籍”(book)中, 用以引入应出现在末尾的材料(如参考文献, 索引等), 本命令会关闭`\chapter`命令的章计数器.

`\backslash` [m] 产生`\`.

`\bar{x}` [m] 产生数学变量上的重音号: `\bar{a} = ā`.

`\Bar{x}` [m][a] 本命令与`\bar`同样使用, 不过当出现 \mathcal{A} 的多重数学重音号时, 本命令会自动调节位置.

`\baselineskip` 在段落内部的行间距, 每种字体有它自己的行间距, 可以用命令`\setlength`为其指定新值(一个弹性长度):

`\setlength{\baselineskip}{12pt plus2pt minus1pt}`

`\baselinestretch` 正常值等于1的一个倍数, 内部长度`\baselineskip`乘以这个倍数后就得到实际上被使用的行间距, 可用以下命令改变它的值:

`\renewcommand{\baselinestretch}{倍数}`

不过新的值要在遇到改变字体的命令后才能生效!

`\begin{环境名}` 进入环境名指定的环境, 这个命令必须与`\end{环境名}`配对以退出此环境.

`\begin{abstract}` 进入“摘要”(abstract)环境以生成一个摘要, 当文件类是“文章”(article)时, 选用的字体是`\small`, 选用的环境是“引文”(quotation), 当文件类是“报告”(report)时, 摘要被单独排成一页, 且使用正常大小的字体与行宽, 在上述两种情形里, 标题`Abstract`都位于上部居中位置.

`\begin{align}` [a] 本命令进入行间数学模式, 生成一组对齐的数学公式, 以 `\\` 标志行的结束, 每一行被第一、三、五、... 个 `&` 号分成几列, 分别对齐, 如果不选用本命令的 `*` 形式, 则每一行都会得到一个公式编号.

`\begin{alignat}{数}` [a] 本命令的作用类似 `align` 环境, 不过它不在列对之间插入额外的空白, 输入的参数等于列对的个数, 也就是说, $数 = (1 + n_{\&})/2$, 其中 $n_{\&}$ 是每行的 `&` 的个数, 列对之间可以用手工方式插入空白, 尤其当列对的左半部为空时.

`\begin{aligned}` [竖直位置] [m][a] 本命令的作用类似于 `align` 环境, 不过它在数学环境里自成一个单元, 可选参数竖直位置规定了它与相邻单元间的位置关系: 它的取值可为 `t` 或 `b`, 分别表示按顶行或底行对齐, 没有参数则表示中间对齐.

`\begin{appendix}` 进入“附录”环境以生成附录, 在文件类为“文章”(book)或“报告”(report)时, 章号计数器被置零, 附录中相应的节号或章号以大写字母计数.

`\begin{array}` [竖直位置]{列格式} [m] 进入 `array` 环境以生成矩阵或数学模式内的阵列, 列格式由每列对应一个格式字符所组成, 例如 `\begin{array}{lcr}` 产生一个含 3 列的阵列, 其中第一列左对齐, 第二列中间对齐, 第三列右对齐. 可选参数竖直位置规定了这个阵列与同一行里其它单元间的位置关系: `t` 表示顶上的行对齐, `b` 表示底部的行对齐, 没有参数则是中间对齐. 参见 `\begin{tabular}`.

`\begin{center}` 进入 `center` 环境, 以命令 `\\` 表示行的结束, 每行均居中, 参见 `\centering`.

`\begin{命令名}` 大多数声明性的命令, 例如改变字型或字的大小的声明等, 都能被作为一个环境的名称, 例如 `\begin{small}` 与 `\end{small}` 之间的内容将会用 `\small` 的字体打印.

`\begin{alltt}` 在输入了 `alltt` 宏包后, 此环境把除 `\ { }` 以外的字符都用打印机字体打印, 包括其它特殊字符, 且换行符仍起换行的作用, 这样就使各种命令在此环境中仍然有效.

`\begin{bmatrix}` [m][a] 类似于 `matrix` 环境, 但两边添加了方括号 `[]`.

`\begin{Bmatrix}` [m][a] 类似于 `matrix` 环境, 但两边添加了花括号 `{ }`.

`\begin{cases}` [m][a] 此环境的各行以 `\\` 结束, 各行均左对齐, 每行中可以 `&` 号另起一列, 并相互对齐, 最后在左边以花括号括起来, 并且竖直居中.

`\begin{description}` 进入 `description` 环境以生成一个带有缩进的罗列, 其标签的内容由命令 `\item[标签]` 规定.

`\begin{displaymath}` 从文本模式进入行间数学模式, 使数学公式单独占据一行, 其作用同`\[`.

`\begin{document}` 进入文本文件的最外层环境, 此命令也是前言部分的结束标志, 此命令是每个 L^AT_EX 文件不可缺少的, 与其配对的是`\end{document}`.

`\begin{enumerate}` 进入 `enumerate` 环境以生成一个带缩进以及编号的罗列, 编号数字的格式与嵌套深度有关, 最外层的编号是一个阿拉伯数字, 每遇到一个`\item`就增加 1.

`\begin{eqnarray}` 从文本模式进入行间数学模式, 生成按`{rcl}`形式分 3 列对齐的一组公式或一个多行公式, 公式的每一行均以`\\`结束, 行内各列以`&`分隔, 每个不含命令`\nonumber`的行都会得到一个连续的编号.

`\begin{eqnarray*}` 类似于 `eqnarray` 环境, 只是没有编号.

`\begin{equation}` 从文本模式进入行间数学模式, 生成单独占一行的数学公式, 并得到一个自动生成的编号.

`\begin{falign}` 类似于 `align` 环境, 只是在列对之间插入空白, 使它水平伸展到版面的宽度.

`\begin{figure}[位置]` 把图形嵌入文本的浮动环境, 可选参数位置可以是字母 `h`, `t`, `b`, `p` 的任意组合, 其中 `h` 表示当前位置, `t` 表示页面顶部, `b` 表示页面底部, `p` 表示单独一页, 默认值是 `tbp`, 在 L^AT_EX 2_ε 中还可以使用字母 `!` 表示取消对浮动所加的一些限制.

`\begin{figure*}[位置]` 类似于 `figure` 环境, 不过它在由可选项 `twocolumn` 或命令 `\twocolumn` 规定的双栏页面格式中占据两栏的宽度, 不像标准形式的 `figure` 只占据一栏的宽度.

`\begin{filecontents}{文件名} [p]` 本环境只能出现在`\documentclass`之前, 如果文件名所指示的文件不存在, 则创建此文件, 并把环境里的文本不作更动地复制到此文件内, 同时在文件的头部用注解的形式说明其来源, 这条命令用来传递一些非标准的附加文件, 把这些文件的内容附加在主文件的头部一起传递到别处, 在编译主文件时就能创建这些必要的附加文件, 而且可被立即引用, 如果同名文件已经存在, 则会发出一个警告信息, 并不改动已有文件.

`\begin{filecontents*}{文件名} [p]` 作用类似于 `filecontents` 环境, 只是不在头部附加来源信息, 因此生成新文件的内容和环境中的内容完全相同.

`\begin{flushleft}` 进入 `flushleft` 环境, 使得其中的每一行只保持左边界对齐, 不保持右边界对齐, 参见 `\raggedright`.

`\begin{flushright}` 进入 `flushright` 环境, 使得其中的每一行只保持右边界对齐, 不保持左边界对齐, 参见 `\raggedleft`.

`\begin{gather}` [a] 进入行间数学模式, 生成各行均居中的多行数学公式, 而不考虑竖直对齐, 每行均以`\\`结束, 而且会得到一个编号, 本命令的`*`形式除不给公式编号外, 其余都相同.

`\begin{gathered}` [竖直位置] [m][a] 本命令的作用类似于`gather`环境, 不过它在数学环境里自成一个单元, 可选参数竖直位置规定了它与相邻单元间的位置关系: 它的取值可为`t`或`b`, 分别表示按顶行或底行对齐, 没有参数则表示中间对齐.

`\begin{itemize}` 进入`itemize`环境, 生成一个带缩进以及标签的罗列, 标签的类型与嵌套的深度有关: 最外层的标签是由`\item`命令产生的`•`号.

`\begin{letter}`{收信人} 当文件类是“信件”(letter)时创建一封信, 收信人中的内容包括收信人的姓名与地址等, 用`\\`表示换行.

`\begin{list}`{标准标签格式}{列表声明} 进入一般列表环境, `\item`命令所产生的标签格式由标准标签格式规定, 而列表声明则包含一系列重定义各项列表参数的命令.

`\begin{lrbox}`{\盒子名} 本命令的作用与`\sbox`相似, 在本命令之前应该先用命令`\newsavebox{\盒子名}`创建一个名为\盒子名的LR盒子, 然后本环境所包含的内容将被存储在这个盒子里, 使用命令`\usebox{\盒子名}`即可打印此盒子的内容, 且可反复使用.

`\begin{math}` 从文本模式转为行内数学模式, 以在一行文本内插入一个数学公式, 其效果同`\(`或文本模式中的`$`.

`\begin{matrix}` [m][a] 其作用类似`array`环境, 不过列格式的声明可被忽略, 在没有列格式声明的情况下, 可以产生不超过10个居中的列, 否则就必须改变计数器`MaxMatrixCols`的值, 类似的环境还有`pmatrix`, `bmatrix`, `Bmatrix`, `vmatrix`, `Vmatrix`, 它们分别用`()`, `[]`, `{}`, `||`, `|||`符号把阵列括起来.

`\begin{minipage}`[竖直位置][高度][内部位置]{宽度} 把文本内容排版在一个指定宽度的“小页”上, 可选项竖直位置确定了此小页相对于所处环境的位置: `t`表示顶行对齐, `b`表示底行对齐, 没有参数表示中间对齐, 另两个可选参数是: 高度规定总高度, 内部位置规定文本在小页内部的位置: `t`表示顶部, `b`表示底部, `c`表示居中, `s`表示扩展成充满整个空间, 默认值是竖直位置的取值, 高度中可包含参数`\height`, `\depth`, `\width`以及`\totalheight`.

`\begin{multicols}`{栏数}[标题][预留高度] 这个环境是由工具包`multicol`提供的, 由此处起, 页面被分成栏数所规定的几栏, 标题作为横跨的一栏打印在顶部, 如果当前页的剩余高度小于`\premulticols`或小于可选参数预留高度, 就会自动换页, 在末尾时是否生成新页取决于剩余高度与`\postmulticols`的

关系, 所有页面的各栏都自动保持同样高度, 栏间距以及栏间分隔线的厚度由 `\columnsep` 与 `\columnseprule` 确定.

`\begin{multline}` [a] 进入行间数学模式以生成一个分拆成数行的数学公式, 公式的第一行向左对齐, 最后一行向右对齐, 中间各行均居中, 用 `\\` 表示换行, 利用 `\shoveleft{数学式}` 或 `\shoveright{数学式}` 可使数学式占据一行, 并向左或向右对齐. 类的选项 `reqno` (默认值) 或 `leqno` 决定了公式编号放在最后一行的右边或第一行的左边, 本环境对应的 * 形式没有公式编号.

`\begin{note}` 在文件类“投影片”(slide)中, 本环境生成当前投影片的一个注记, 在 L^AT_EX 2.09 中这是一个黑白片, 注记的编号是在当前投影片编号的后面用短划再加一个顺序号, 例如: 8-1, 8-2 等.

`\begin{overlay}` 在文件类“投影片”(slide)中, 本环境生成当前投影片的一个覆盖片, 覆盖片的编号是在当前投影片编号的后面用短划再加一个小写字母, 例如: 3-a, 3-b 等.

`\begin{picture}`(宽度, 高度)(*x* 坐标, *y* 坐标) 本环境生成一个具有指定宽度与高度的图形, 这里的(*x* 坐标, *y* 坐标)是图形左下角参考点的坐标, 使用的长度单位是已被定义的 `\unitlength` (默认值是 1pt).

`\begin{quotation}` 进入 quotation 环境, 其中的文本相对于正常边界从两边缩进, 环境内部的段落首行还有额外的缩进.

`\begin{quote}` 类似于 quotation 环境, 不过环境内部的段落首行没有额外的缩进, 但段落之间有额外的间距.

`\begin{slide}` 在文件类“投影片”(slide)中, 本环境是生成投影片的主环境.

`\begin{sloppypar}` 在这个环境里, 词与词之间的间隔允许大于通常的值, 使得一个段落可以分拆成更多的行, 参见 `\sloppy`, 与此相反的命令是 `\fussy`.

`\begin{split}` [m][a] 此环境出现在数学模式中, 其作用是使得一个公式被分拆成数行, 并用 `\\` 标注换行, 各行按 & 号竖直对齐, 其公式编号由外部环境给出, 当类选项为 `centertags` (默认值) 时, 编号竖直居中, 为 `tbtags` 时, 还要根据类选项是 `reqno` (默认值) 还是 `leqno`, 分别置放于最后一行的右边或第一行的左边.

`\begin{subarray}`{位置}{第一行`\\`..`\\`最后一行} [m][a] 类似于 `\substack`, 可在数学模式里生成占据多行的上标或下标, 位置的取值可以是 `c` (居中对齐) 或 `l` (靠左对齐).

`\begin{subequations}` [a] 在这个环境里的公式编号具有一个主要数字再附加小写字母的形式, 例如: 7a, 7b, 7c 等.

`\begin{tabbing}` 进入 tabbing 环境, 使得特殊的制表位命令生效: `\=` 设立一个

制表位, \> 进到下一个制表位, \< 退到上一个制表位, \\ 换行, \+ 使左边界右移一个制表位, \- 使左边界左移一个制表位.

`\begin{table}`[位置] 把表格嵌入文本的浮动环境, 可选参数位置可以是字母 h, t, b, p 的任意组合, 其中 h 表示当前位置, t 表示页面顶部, b 表示页面底部, p 表示单独一页, 默认值是 tbp, 在 L^AT_EX 2_ε 中还可以使用字母 ! 表示取消对浮动所加的一些限制.

`\begin{table*}`[位置] 类似于 table 环境, 不过它在由可选项 twocolumn 或命令 `\twocolumn` 规定的双栏页面格式中占据两栏的宽度, 不像标准形式的 table 只占据一栏的宽度.

`\begin{tabular}`[竖直位置]{列格式} 进入 tabular 环境以生成一个表格, 列格式由每列对应一个格式字符所组成: c 表示居中, l 表示左对齐, r 表示右对齐, p{宽度} 表示一个具有指定宽度的列, 其中的内容可占据多行, 此外, @{文本} 可以出现在任意两个上述的列格式之间, 其中的文本将被插入每一行的同一位置, 字符 | 表示贯穿各行的一条竖直线.

可选参数竖直位置规定了这个表格与外部基线间的位置关系: t 表示顶上的行与基线对齐, b 表示底部的行对齐, 没有参数则是中间对齐.

表格的各行以 \\ 分隔, 同一行的各列则以 & 分隔.

`\begin{tabular*}`{宽度}[竖直位置]{列格式} 类似于 `\begin{tabular}`, 不过表格的总宽度由参数宽度给出, 为达到这个宽度必须在列的间隔中含有弹性长度, 为此可在列格式的某些地方插入 @{\extracolsep\fill}.

`\begin{thebibliography}`{标签样本} 进入生成参考文献的环境, 标签样本是可能出现的最长的标签, 以命令 `\bibitem` 作为一个条目的开始, 此命令为这个条目产生一个标签, 以后各行都缩进标签样本所占据的宽度.

`\begin{theindex}` 此环境以双列格式生成一个索引纪录, 每个条目的起首是命令 `\item`, `\subitem`, `\subsubitem` 再后接关键词, 或者是 `\indexspace`.

`\begin{定理环境名}`[附加标题] 此环境调用一个已被用户用命令 `\newtheorem` 定义过的定理类结构, 这里的定理环境名就是命令 `\newtheorem` 的第一个参数, 一般命名为 theorem, proposition, axiom 等. 附加标题 (例如: Fermat's Theorem) 则用圆括号 () 括起来接在定理名称及编号的后面.

`\begin{titlepage}` 此环境产生一个没有页码的标题页, 用户对此页面的构成有完全的控制.

`\begin{trivlist}` 此环境生成一个既无标签样本又无格式声明的平凡列表, 参数 `\leftmargin`, `\labelwidth`, `\itemsep` 都被置为 0pt, 而 `\listparindent = \parindent`, `\parsep = \parskip`.

`\begin{verbatim}` 此环境内的文本将被一字不差地按输入的样式打印出来, 包括空白行、换行、注解等全都原原本本被打印. (注意: 其中不能含汉字)

`\begin{verbatim*}` 类似于 `verbatim` 环境, 只是文中的空格被打印成 `\`.

`\begin{verse}` 用于写诗歌、韵文等的环境, 用 `\\` 提示换行, 而段与段用空白行分隔.

`\begin{vmatrix}` [m][a] 类似于 `matrix` 环境, 不过两端用 `||` 括起来.

`\begin{Vmatrix}` [m][a] 类似于 `matrix` 环境, 不过两端用 `|||` 括起来.

`\belowdisplayskip` [m] 一个长的单列公式与后面的文本行之间的竖直留空, 可用 `\setlength` 为其指定新值, 例如: 命令

`\setlength{\belowdisplayskip}{\abovedisplayskip}`

使 `\belowdisplayskip` 取与 `\abovedisplayskip` 相同的值, 参见 `\abovedisplayskip`.

`\belowdisplayshortskip` [m] 一个短的单列公式与后面的文本行之间的竖直留空, 可用 `\setlength` 为其指定新值 (参见上一条目).

`\beta` [m] 产生 β .

`\bezier{点数}(x_1, y_1)(x_2, y_2)(x_3, y_3)` 本命令应被用于 `picture` 环境内, 生成一条端点为 (x_1, y_1) , (x_3, y_3) 的二次 Bézier 曲线, 以 (x_2, y_2) 作为控制点, 此曲线由 (点数 + 1) 个点构成, 本命令类似于 `\qbezier`, 不过后者的点数是自动产生的.

`\bf` 转变为罗马族, 直立形状, 黑体系列 (**Roman, upright, bold**) 的字体属性.

`\bfdefault` 定义了由命令 `\bfseries` 选取的字体系列, 可用 `\renewcommand` 予以重定义:

`\renewcommand{\bfdefault}{b}`

`\bfseries` 本声明不改变当前字体的族与形状, 但转变成 **bold** 序列.

`\bibitem[标签]{引用词}` 条目文本 本命令应被用于 `thebibliography` 环境内, 生成一个参考文献条目, 当正文中引用这个文献时, 以引用词作为命令 `\cite` 的参数, 此时 `\cite` 命令的位置将被带方括号的可选项 [标签] 所取代, 在默认的情形则被一个带方括号的 [标号] 所取代.

`\bibliography{文件名}` 利用程序 `BIBTEX` 生成参考文献: 文件名是包含被搜索的文献数据库的一个或多个文件的根文件名.

`\bibliographystyle{格式}` 在与程序 `BIBTEX` 配合使用时, 本命令选取文献条目的打印格式, 格式的选取可以是 `plain`, `unsrt`, `alpha`, `abbrv`, 默认的是第一个, 也可以有其它非标准的格式.

`\bibname` 在文件类为 “书籍” (`book`) 或 “报告” (`report`) 时, 本命令含有参考文献

的标题, 在英文中是 ‘Bibliography’, 可根据不同的语言加以修改.

`\big` 定界符 [m] 产生比正常大但比 `\Big` 小的定界符, 例如: `\big(= (`.

`\Big` 定界符 [m] 产生比 `\big` 大但比 `\bigg` 小的定界符, 例如: `\Big[= [`.

`\bigcap` [m] 产生 \cap .

`\bigcirc` [m] 产生 \bigcirc .

`\bigcup` [m] 产生 \cup .

`\bigg` 定界符 [m] 产生比 `\Big` 大但比 `\Bigg` 小的定界符, 例如: `\bigg| = |`.

`\Bigg` 定界符 [m] 最大的定界符, 例如: `\Bigg\langle = \langle`.

`\biggl` 定界符 [m] 类似于 `\bigg`, 但它也是左定界符.

`\Biggl` 定界符 [m] 类似于 `\Bigg`, 但它也是左定界符.

`\biggm` 定界符 [m] 类似于 `\bigg`, 但它与两边的符号有更大的留空(作为关系算子).

`\Biggm` 定界符 [m] 类似于 `\Bigg`, 但它与两边的符号有更大的留空(作为关系算子).

`\biggr` 定界符 [m] 类似于 `\bigg`, 但它也是右定界符.

`\Biggr` 定界符 [m] 类似于 `\Bigg`, 但它也是右定界符.

`\bigl` 定界符 [m] 类似于 `\big`, 但它也是左定界符.

`\Bigl` 定界符 [m] 类似于 `\Big`, 但它也是左定界符.

`\bigm` 定界符 [m] 类似于 `\big`, 但它与两边的符号有更大的留空(作为关系算子).

`\Bigm` 定界符 [m] 类似于 `\Big`, 但它与两边的符号有更大的留空(作为关系算子).

`\bigodot` [m] 产生 \odot .

`\bigoplus` [m] 产生 \oplus .

`\bigotimes` [m] 产生 \otimes .

`\bigr` 定界符 [m] 类似于 `\big`, 但它也是右定界符.

`\Bigr` 定界符 [m] 类似于 `\Big`, 但它也是右定界符.

`\bigtriangledown` [m] 产生 ∇ .

`\bigtriangleup` [m] 产生 \triangle .

`\bigskip` 插入一个值为 `\bigskipamount` 的大的竖直间隔, 参见 `\medskip` 以及 `\smallskip`.

`\bigskipamount` 用于 `\bigskip` 的竖直间隔的标准值, 可用命令 `\setlength` 加以改变:

`\setlength{\bigskipamount}{6ex plus 1.5ex minus 2ex}`

`\bigsqcup` [m] 产生 \sqcup .

`\biguplus [m]` 产生 \biguplus .

`\bigvee [m]` 产生 \bigvee .

`\bigwedge [m]` 产生 \bigwedge .

`\binom{上部}{下部} [m][a]` 产生组合数: $\binom{n}{k}$.

`\bmod [m]` 产生函数名 'mod', 例如: $a \bmod b = a \text{ mod } b$.

`\boldmath` 转移到数学模式的黑体字, 请注意: 本命令必须在进入数学模式前出现在文本模式中, 如果只需把公式的一部分转成黑体, 则也可使用命令 `\mbox{\boldmath$...$}` 以暂时进入文本模式.

`\boldsymbol{符号} [m][a]` 当宏包 `amsmath` 或 `amsbsy` 之一被读入后, 本命令可把符号打印成黑体, 与 `\mathbf` 不同, 本命令对数学符号以及小写希腊字母也有效.

`\bot [m]` 产生 \bot .

`\botfigrule` 本命令仅当一个浮动环境出现在页底时被执行, 它通常被定义为空, 但可被重定义以在页面的正文与浮动部分之间画一条直线, 请注意这条命令不应该产生额外的竖直空间, 例如:

```
\renewcommand{\botfigrule}{\vspace*{-.4pt}
\rule{\columnwidth}{.4pt}}
```

`\bottomfraction` 页面中可供底部的浮动单元使用的最大部分, 这是一个十进小数, 可用以下命令重置其值:

```
\renewcommand{\bottomfraction}{0.5}
```

`bottomnumber` 此计数器记录了可出现在一个页面底部的浮动单元最大个数, 可用命令 `\setcounter{bottomnumber}{3}` 为其重置新值.

`\bowtie [m]` 产生 \bowtie .

`\Box [m]` 产生 \Box .

`\Boxed{公式} [m][a]` 把公式框起来.

`\breve{x} [m]` 产生数学变量 x 的重音号: $\breve{a} = \check{a}$.

`\Breve{x} [m][a]` 类似于 `\breve`, 但当遇到多重 \mathcal{S} -L^AT_EX 重音号时, 会自动调整其位置.

`\bullet [m]` 产生 \bullet .

`\c{x}` 在 x 的底下产生一个变音号 cedilla: $\c{C} = \mathfrak{C}$.

`\cal [m]` 是 L^AT_EX 2.09 中使用的声明, 在数学模式内选取书写体 (calligraphic), 在 L^AT_EX 2_ε 中已被 `\mathcal` 取代.

`\cap [m]` 产生 \cap .

`\caption[简短形式]{标题文本}` 在浮动环境 `figure` 或 `table` 内生成一个含有编号以及标题文本的标题, 可选项简短形式是用来在目录中代替标题文本的.

`\captions语言` 如果输入了宏包 `esperant`, `german` 或者 `babel`, 此命令用于在多语言环境中重定义一些特定的标题, 例如 ‘Chapter’, ‘Contents’ 等, 本命令通常是命令 `\selectlanguage` 定义的一部分.

`\cc{表}` 在文件类 “信件” (`letter`) 中, 本命令在信末生成 ‘cc:’, 后接一个姓名的表.

`\ccname` 在文件类 “信件” (`letter`) 中, 本命令含有 `\cc` 命令所打印出来的文字, 在英文环境里是 ‘cc’, 但可根据使用的语言重新定义.

`\cdot [m]` 产生 \cdot .

`\cdots [m]` 产生 \cdots .

`\centering` 转换成每行皆居中的格式, 用 `\\` 标志换行, 参见 `\begin{center}`.

`\centerline{文本}` 让文本居中成一行的 T_EX 命令.

`\cfrac[位置]{上部}{下部} [m][a]` 本命令产生一个连分数, 可选项位置可取值 1 或 r, 分别表示分子向左或右对齐, 默认是居中对齐.

`\chapter[短名]{章名}` 另起一页开始新的一章, 以一个自动产生的编号加上章名作为章的标题, 如果给出了可选项短名, 那么它将在目录以及书眉上取代原有的章名.

`\chapter*{章名}` 另起一页开始新的一章, 以章名作为章的标题, 但没有编号, 而且章名也不出现在目录中.

`\chaptername` 本命令包含章的标题, 在英文环境中是 ‘Chapter’, 可被重新定义以适应其它语言.

`\check{x} [m]` 在数学变量 x 上产生重音号: `\check{a} = \check{a}` .

`\Check{x} [m][a]` 类似于 `\check`, 但是在出现多重 \mathcal{A} -L^AT_EX 重音号时, 会自动调整位置.

`\CheckCommand{\命令名}[变量数][可选参数]{定义}` 检查命令 `\命令名` 的当前定义是否与料想的定义一致, 若不一致, 就会发出一个出错信息, 此命令用于保证重要的命令不会被别的宏包修改.

`\CheckCommand*{\命令名}[变量数][可选参数]{定义}` 类似于 `\CheckCommand`, 但是要求被检查的命令是短的命令, 即在定义中不能出现新的段落, 也就是说, 不能有 `\par` 及空白行.

`\chi [m]` 产生 χ .

`\circ [m]` 产生 \circ .

`\circle{直径}` 在图形 (picture) 环境里产生一个指定直径的圆周, 本命令应被用在命令 `\put` 或 `\multiput` 中作为参数.

`\circle*{直径}` 类似于 `circle`, 不过产生一个实心圆.

`\cite[附注]{引用词}` 利用引用词在正文中产生对参考文献标签的引用, 可选参数附注被添加在标签的后面.

`\ClassError{类名}{出错信息}{帮助}` [p] 本命令只能出现在类文件中, 它向监视器以及纪录文件发出以类名为标号的出错信息, 中断进程并等待用户的反应, 如果用户键入 H<回车>, 就把帮助打印出来, 在出错信息和帮助中都可用 `\MessageBreak` 命令换行, 用 `\space` 强制产生空格, 并可用 `\protect` 冠在一个命令前面, 使得这个命令不被解释执行而是把它的名字打印出来.

`\ClassInfo{类名}{信息}` [p] 类似于 `\ClassWarningNoLine`, 不过信息不在监视器上显示, 只是写入纪录文件.

`\ClassWarning{类名}{警告信息}` [p] 本命令只能出现在类文件中, 它向监视器以及纪录文件发出以类名为标号的警告信息, 同时给出输入文件的当前行号, 并继续进行下去, 警告信息可按 `\ClassError` 同样的方法格式化.

`\ClassWarningNoLine{类名}{警告信息}` [p] 类似于 `\ClassWarning`, 只是不打印当前行号.

`\cleardoublepage` 结束当前页, 并把所有尚未处理的浮动单元输出到一个或几个浮动页上, 接在后面的页面将是具有奇数页码的右页.

`\clearpage` 结束当前页, 并把所有尚未处理的浮动单元输出到一个或几个浮动页上.

`\cline{n-m}` 在 `tabular` 环境里生成一条从第 n 列到第 m 列的水平线, 例如:

`\cline{2-5}.`

`\closing{问候语}` 信件 (letter) 环境里正文结束后, 问候语用于收尾部分.

`\clubsuit` [m] 产生 ♣.

`\color` 色彩 必须先输入宏包 `color` 才能使用本命令, 它是选定某种色彩的声明, 以后的文本就用这种颜色打印, 它的作用范围持续到所在环境的结束或者遇到另一条 `\color` 命令为止. 色彩可以是预定义的或已用 `\definecolor` 定义过的颜色名, 也可具有 [模式]{数据} 的形式, 其中参数数据的意义同 `\definecolor`, 例如:

`\color[rgb]{0.5,0.5,0}` `\color{magenta}`

`\colorbox` 色彩{文本} 必须先输入宏包 `color` 才能使用本命令, 文本生成一个 LR 盒子, 并以指定的色彩作为盒子的背景色, 色彩的定义同 `\color`.

`\columnsep` 规定了双栏页面格式中的栏间距值, 可用 `\setlength` 重置:

`\setlength{\columnsep}{1pt}`

`\columnseprule` 规定了双栏页面格式中的栏间竖直线的厚度, 可用 `\setlength` 重置: `\setlength{\columnseprule}{1pt}`

`\cong [m]` 产生 \cong .

`\contentsline{章节类型}{\numberline{编号}标题}{页码}` 本命令出现在 `toc` 文件内, 每一个命令对应目录中的一个条目. 当 T_EX 遇到 `\tablecontents` 命令时, 才把这个文件读入, 因此用户可根据需要利用文本编辑器修改或添加 `toc` 文件中的这种命令. 章节类型是指章节的层次, 例如 `section`, 编号是此章节的编号 (例如 2.3), 页码是正文中出现的页码.

`\contentsname` 本命令包含目录的标题, 在英文环境里是 'Contents', 可根据需要重新定义以与所使用的语言一致.

`\coprod [m]` 产生 \coprod .

`\copyright` 产生 ©.

`\cos [m]` 本命令生成数学公式里的函数名 'cos'.

`\cosh [m]` 本命令生成数学公式里的函数名 'cosh'.

`\cot [m]` 本命令生成数学公式里的函数名 'cot'.

`\coth [m]` 本命令生成数学公式里的函数名 'coth'.

`\csc [m]` 本命令生成数学公式里的函数名 'csc'.

`\cup [m]` 产生 \cup .

`\CurrentOption [p]` 本命令包含当前正被处理的可选项名称, 它仅能用于可选项的定义中, 特别是默认可选项的定义中.

`\d{x}` 产生置于字母底下的一点: `\d{o} = o`.

`\dag` 产生 †.

`\dagger [m]` 产生 †.

`\dashbox{虚线长}(宽度, 高度)[位置]{文本}` 本命令仅能用于 `picture` 环境中, 生成一个带虚线框的盒子, 具有指定的宽度、高度以及虚线长, 可选参数位置是指文本在盒子内部的位置, 它们可以是居左 (l), 居右 (r), 居顶 (t) 或居底 (b), 也可以是两者的组合, 如 `lt`, 默认值是居中, 本命令被用作 `\put` 或 `\multiput` 命令的参数.

`\dashv [m]` 产生 \dashv .

`\date{日期}` 1. `\maketitle` 命令通常在目录页中打印当前日期, 用户可利用这条命令打印任何文本.

2. 在“信件”内打印指定的日期以取代自动打印当前日期.

`\date语言` 如果输入了宏包 `esperant`, `german` 或者 `babel`, 此命令用于在多语言

环境中重定义 `\today` 命令, 例如 `\dateUSenglish`, `\dateenglish` 等, 本命令通常是命令 `\selectlanguage` 定义的一部分.

`\dbinom{上部}{下部} [m][a]` 类似于 `\binom`, 生成一个组合数, 不过其中字母的尺寸是 `\displaystyle`.

`\dblfigrule` 本命令仅当一个双栏的浮动环境出现在页顶时被执行, 它通常被定义为空, 但可被重定义以在页面的正文与浮动部分之间画一条直线, 请注意这条命令不应该产生额外的垂直空间, 例如:

```
\renewcommand{\dblfigrule}{\vspace*{-.4pt}
\rule{\textwidth}{.4pt}}
```

`\dblfloatpagefraction` 在双栏页面格式中, 一个浮动页至少必须占据一页的多少部分才能再生成一个新的页, 这是一个十进小数, 可用以下命令指定它的新值:

```
\renewcommand{\dblfloatpagefraction}{0.8}
```

`\dblfloatsep` 在双栏页面格式中, 上下两个跨栏的浮动单元间的垂直间隔, 可用以下命令指定它的新值:

```
\setlength{\dblfloatsep}{12pt plus2pt minus4pt}
```

`\dbltextfloatsep` 在双栏页面格式中, 一个位于页顶的浮动单元与在它下面的正文间的垂直间隔, 可用 `\setlength` 命令指定它的新值.

`\dbltopfraction` 在双栏页面格式中, 一个页面可被位于顶部的跨栏浮动单元占据的最大部分, 这是一个十进小数, 可用以下命令指定它的新值:

```
\renewcommand{\dbltopfraction}{0.3}
```

`dbltopnumber` 在双栏页面格式中, 一个页面顶部可出现的跨栏浮动单元的最大个数, 可用以下命令指定它的新值:

```
\setcounter{dbltopnumber}{3}
```

`\ddag` 产生 ‡.

`\ddagger [m]` 产生 ‡.

`\ddot{x} [m]` 在数学公式中产生一个两点重音号: `\ddot{a} = ä`.

`\DeclareErrorFont{编码}{族}{系列}{形状}{尺寸} [p]` 如果找不到有效的字体, 甚至连 `\DeclareFontSubstitution` 所给出的替代字体也找不到, 那么本命令声明的字体是最后的救星, 例如:

```
\DeclareErrorFont{OT1}{cmr}{m}{n}{10}
```

`\DeclareFixedFont{\命令名}{编码}{族}{系列}{形状}{尺寸} [p]` 把 `\命令名` 定义为选取这条命令所规定的字体的声明.

`\DeclareFontEncoding{编码}{文本模式命令}{数学模式命令} [p]` 声明创立以

编码命名的字体编码体系, 文本模式命令是转换成文本模式时需要执行的命令序列, 同理, 数学模式命令是转换成数学模式时需要执行的命令序列, 编码有: OT1 (Knuth 创建的文本字体, 如 cmr10), OT2 (华盛顿大学的斯拉夫字体, 如 wncyr10), T1 (DC 字体, 如 dcr10), OML (T_EX 数学字母字体, 如 cmml10), OMS (T_EX 数学符号字体, 如 cmsy10), OMX (T_EX 数学扩充字体, 如 cmex10), U (未知编码), 例如:

```
\DeclareFontEncoding{OT1}{}{}
```

`\DeclareFontEncodingDefaults{文本模式命令}{数学模式命令}` [p] 声明对所有字体编码体系都适用的进入文本或数学模式时需要执行的命令序列, 具体编码体系的附加命令序列都接在后面执行.

`\DeclareFontFamily{编码}{族名}{命令序列}` [p] 声明创立以族名命名的新字体族, 以已有定义的编码作为编码体系, 每当这个字体族被选取时, 就要先执行指定的命令序列.

`\DeclareFontShape{编码}{族}{系列}{形状}{定义}{命令序列}` [p] 本命令把外部的字库文件名与指定的字体属性相关连, 在定义中把字体尺寸与字库名相联系, 每当这种字体被选用时, 就要先执行指定的命令序列.

`\DeclareFontSubstitution{编码}{族}{系列}{形状}` [p] 如果对应某种字体属性的有效字体不存在, 就用本命令指定的字体属性来替代, 替代次序是: 形状、系列、族, 编码不能替代. 例如:

```
\DeclareFontSubstitution{U}{cmr}{m}{n}
```

`\DeclareGraphicsExtensions{扩展名表}` [p] 建立可被 `\includegraphics` 命令以及 `graphics`, `graphicx` 宏包输入的图形文件的默认扩展名表, 扩展名表用逗号分隔, 如 `eps`, `ps` 等.

`\DeclareGraphicsRule{扩展名}{类型}{信息文件扩展名}{命令}` [p] 把图形文件的扩展名与图形文件的类型、含有图框尺寸的信息文件扩展名以及一个操作命令相关连, 执行此命令后图形方能被输入, 例如:

```
\DeclareGraphicsRule{.eps.gz}{eps}{.eps.bb}{'gunzip -c #1}
```

其中的命令必须以 ' 为前缀, #1 代表被处理的文件名.

`\DeclareMathAccent{\命令}{类型}{符号字体}{编号}` [p] 声明 `\命令` 为数学重音号命令, 打印内部名称为符号字体中的具有指定编号的字符, 类型可以是 `\mathord` 或 `\mathalpha`, 在后一情形, 符号随着数学字体的变化而变化, 例如:

```
\DeclareMathAccent{\acute}{\mathalpha}{operators}{19}
```

`\DeclareMathAlphabet{\命令}{编码}{族}{系列}{形状}` [p] 把 `\命令` 定义为数

学模式下设置字体的命令, 对所有的数学变体(除了正常字体(normal)外, 目前仅有的变体是bold, 用\boldmath选取), 选取的是同样的具有指定属性的字体, 如果想对不同的变体使用不同字体的话, 还需要用\SetMathAlphabet个别地定义. 当形状属性留空时, 本命令虽然被创立了, 但对所有的变体都是未经定义的, 仍需要用\SetMathAlphabet对各种变体分别予以定义, 例:

```
\DeclareMathAlphabet{\mathsl}{OT1}{cmr}{m}{sl}
```

\DeclareMathDelimiter{\命令}{类型}{字体一}{编号}{字体二}{编号} [p] 声明\命令是有两种不同大小的定界符, 小的变体取自内部名称为字体一的具有所给编号的字符, 大的变体取自内部名称为字体二的具有所给编号的字符, 例如:

```
\DeclareMathDelimiter{(){\mathopen}{operators}{40}
{\largesymbols}{0}
```

\DeclareMathOperator{\命令}{函数名} [p][a] 在输入宏包amsopn与amsmath后, \命令定义为数学模式里的命令, 它用直立字体打印函数名, 并有适当的留空, 相应的*形式命令还可以用^与_输入上下界.

\DeclareMathRadical{\命令}{字体一}{编号}{字体二}{编号} [p] 声明\命令是有两种不同大小的数学根式符号, 小的变体取自内部名称为字体一的具有所给编号的字符, 大的变体取自内部名称为字体二的具有所给编号的字符, 例如:

```
\DeclareMathRadical{\sqrtsign}{symbols}{112}{largesymbols}{112}
```

\DeclareMathSizes{正文}{数学正文}{角标}{二级角标} [p] 规定3种数学字体\textstyle, \scriptstyle, \scriptscriptstyle的大小点数, 这里4个参数都是整数, 其单位是pt, 第一个参数正文是正文字体的点数, 后面3个分别对应上述3种数学字体的点数, 例如:

```
\DeclareMathSizes{10}{10}{7}{5}
```

\DeclareMathSymbol{\命令}{类型}{字体}{编号} [p] 声明\命令是一个数学符号, 打印出内部名称为字体的具有所给编号的字符, 它在所有数学字体命令下打印同一个符号, 但对于不同的变体可能有不同的符号字体, 不过这必须由\Set...命令设置成别的属性, 这里的类型可以是: \mathord(通常的符号), \mathop(大型算子, 如 \sum), \mathbin(二元算子, 如 \times), \mathrel(关系算子, 如 \geq), \mathopen(开括号, 如 $($), \mathclose(闭括号, 如 $)$), \mathpunct(标点符号), \mathalpha(字母), 例如:

```
\DeclareMathSymbol{\alpha}{\mathord}{letters}{11}
```

\DeclareMathVersion{变体名} [p] 声明数学字体或符号的一个新变体, 开始时此

变体将使用 `\Declare...` 命令所规定的字体属性, 但它可以被合适的 `\Set...` 命令重新设置, 在正文中可用 `\mathversion{变体名}` 命令选取此变体.

`\DeclareOldFontCommand{\命令}{文本模式命令}{数学模式命令}` [p] 把 `\命令` 定义为字体声明, 文本模式时使用文本模式命令, 数学模式时使用数学模式命令, 本命令的目的是定义与 L^AT_EX 相容的命令, 一般应避免使用, 例:

```
\DeclareOldFontCommand{\it}{\normalfont\itshape}{\mathit}
```

`\DeclareOption{选项}{命令序列}` [p] 在类文件或宏包中, 本命令给出了与选项相关的命令序列, 这些命令是在调用 `\ExecuteOptions` 或 `\ProcessOptions` 时被执行的, 后两条命令执行后, 所有的定义都从内存中清除, 命令序列则被存储在内部命令 `ds@选项` 中.

`\DeclareOption*{命令序列}` [p] 在类文件或宏包中, 给出了与所有未定义的选项相关联的默认命令序列, 命令序列中可使用的特殊命令是 `\CurrentOption` (指选项名) 及 `\OptionNotUsed`, 例如:

```
\DeclareOption*{\InputIfFileExists
  {\CurrentOption.sty}{\OptionNotUsed}}
```

`\DeclareRobustCommand{\命令}[变量数][选项]{定义}` 类似于 `\newcommand` 那样定义或重定义 `\命令`, 只是所得的命令是坚固的: 它可用作其它命令的变量, 而不需在前面冠以 `\protect`.

`\DeclareRobustCommand*{\命令}[变量数][选项]{定义}` 类似于不带星的命令 `\DeclareRobustCommand`, 只是定义必须是短的: 即在定义中不能出现新的段落, 也就是说, 不能有 `\par` 及空白行.

`\DeclareSymbolFont{字体}{编码}{族}{系列}{形状}` [p] 声明具有指定字体属性的字体为符号字体, 其内部名称为第一个参数字体, 此符号字体可用于所有的数学变体, 否则要用 `\SetSymbolFont` 重定义.

`\DeclareSymbolFontAlphabet{\命令}{字体}` [p] 把 `\DeclareSymbolFont` 定义的内部名称为字体的字体声明为 `\命令` 所指的数学字体, 当已经定义了一个合用的数学字体时, 此命令比 `\DeclareMathAlphabet` 更适用, 例如:

```
\DeclareSymbolFontAlphabet{\mathrm}{operators}
```

`\DeclareTextAccent{\命令}{编码}{字符编号}` [p] 当指定的编码处于激活状态时, `\命令` 定义为重音号命令, 它以字体中位于字符编号位置的符号作为重音号, 例如:

```
\DeclareTextAccent{\'}{OT1}{19}
```

`\DeclareTextAccentDefault{\命令}{编码}` [p] 当缺乏激活的编码时, 本命令声明用于重音号命令的指定编码.

`\DeclareTextCommand{\命令}{编码}[变量数][选项]{定义}` [p] 像 `\newcommand` 那样定义 `\命令`, 只是仅当指定的编码处于激活状态时, 定义才有效.

`\DeclareTextCommandDefault{\命令}[变量数][选项]{定义}` [p] 对所有的编码都为 `\命令` 建立一个默认的定义.

`\DeclareTextComposite{\命令}{编码}{字母}{字符编号}` [p] 当指定的编码处于激活状态时, 把 `\命令` 后接字母定义为字体中位于字符编号位置的字符, 于是在 T1 编码, 并且带有重音号的字母有一个单独的字符与之对应时, 就定义了重音号命令的一个作用, 例如:

```
\DeclareTextComposite{\'}{T1}{e}{233}
```

当然这个重音号命令必须用 `\DeclareTextAccent` 或 `\DeclareTextCommand` (带一个参数) 对于指定的编码定义过.

`\DeclareTextCompositeCommand{\命令}{编码}{字母}{定义}` [p] 类似于上述命令 `\DeclareTextComposite`, 只是对于 `\命令` 后接字母的组合可以指定任意的定义.

`\DeclareTextFontCommand{\命令}{定义}` [p] 定义 `\命令` 为指定文本字体的命令, 例如:

```
\DeclareTextFontCommand{\textbf}{\bfseries}
```

`\DeclareTextSymbol{\命令}{编码}{字符编号}` [p] 当指定的编码处于激活状态时, 定义 `\命令` 为打印位于字符编号位置的字符, 例如:

```
\DeclareTextSymbol{\AE}{OT1}{29}
```

`\DeclareTextSymbolDefault{\命令}{编码}` [p] 当缺乏激活的编码时, 本命令声明用于符号命令的指定编码.

`\definecolor{色彩名}{模式}{数据}` 必须先输入宏包 `color` 才有效, 它给色彩名指定一种颜色, 可用的模式为: `rgb` (红, 绿, 蓝), `cmyk` (青, 洋红, 黄, 黑), `gray` (灰) 以及 `named` (已命名的). 数据是一组用逗号分隔的 0 与 1 之间的十进小数串, 表示每个分量的强度. 在 `named` 模式, 数据应取成驱动程序认识的色彩名, 例如:

```
\definecolor{litegrn}{cmyk}{0.25,0,0.75,0}
```

```
\definecolor{brown}{named}{RawSienna}
```

在所有的彩色驱动程序中都被预定义的颜色有: `red`, `green`, `blue`, `yellow`, `cyan`, `magenta`, `black`, `white`.

`\deg` [m] 在数学公式中产生函数名 `'deg'` 的命令.

`\DeleteShortVerb{\字符}` 当标准宏包 `shortvrb` 被输入后, 本命令取消前面的命令 `\MakeShortVerb{\字符}` 的作用, 使字符恢复其本来的意义.

`\Delta` [m] 产生 Δ .

`\delta` [m] 产生 δ .

`\depth` 表示盒子的深度(从基线到底部)的长度参数,它只能用于指定 `\makebox`, `\framebox` 或 `\savebox` 的宽度参数中,或者用于 `\parbox` 及 `minipage` 环境的高度参数中,例如:

`\framebox[20\depth]{text}`

`\det` [m] 在数学公式中产生函数名 'det' 的命令,可带下标作为下界.

`\dfrac{分子}{分母}` [m][a] 类似 `\frac` 产生一个分式,不过是以 `\displaystyle` 的大小.

`\DH` 当 T1 编码被激活时,产生字母 D.

`\dh` 当 T1 编码被激活时,产生字母 d.

`\Diamond` [m] 产生 \diamond .

`\diamond` [m] 产生 \diamond .

`\diamondsuit` [m] 产生 \diamond .

`\dim` [m] 在数学公式中产生函数名 'dim' 的命令.

`\discretionary{前面}{后面}{整体}` 单词内部的可能拆分方案,它可被分成两部分,使前面在行尾,后面在下一行的头,如不被拆开时则打印整体,例如在德文中可能有:

`\discretionary{k-}{k}{ck}`

`\displaybreak[数]` [m][a] 本命令允许在多行数学公式里换页,只要插在 `\\` 之前,数可以取 0-4,按递增次序鼓励换页,如想在多行公式内自动换页,则必须在前言部分发出 `\allowdisplaybreaks` 命令.

`\displaystyle` [m] 在数学模式里使字体大小按 `\displaystyle` 选取.

`\div` [m] 产生 \div .

`\DJ` 当 T1 编码被激活时,产生字母 D.

`\dj` 当 T1 编码被激活时,产生字母 d.

`\documentclass[选项]{类}[版本号]` [p] 通常是 L^AT_EX 2_ε 文件的第一个命令,它确定了文件的整体特征,类的标准取值为:

`article, report, book, letter, slides`

一次只能选取其中之一,选项可在下表中选取,多于一个时可用逗号分隔:

`10pt, 11pt, 12pt,`

`letterpaper, legalpaper, executivepaper,`

`a4paper, a5paper, b5paper, landscape,`

`onecolumn, twocolumn,`

oneside, twoside,
 notitlepage, titlepage,
 leqno, fleqn, openbib,
 draft, final

这些或其它增加的选项都是整体的, 它们对后面由 `\usepackage` 命令输入的宏包都有效,

版本号是形如 `yyyy/mm/dd` 的一个日期, 例如: 1994/08/01, 如果输入类文件的日期比它早, 就会打印一个警告信息.

`\documentstyle[选项]{格式}` [p] 通常是 L^AT_EX 2.09 文件的第一个命令, 它确定了文件的整体特征, 在 L^AT_EX 2_ε 里已被 `\documentclass` 取代, 不过它仍可进入兼容模式, 以处理老式文件.

`\dot{x}` [m] 在数学公式里打印重音号: `\dot{a} = \dot{a}`.

`\Dot{x}` [m][a] 类似于 `\dot`, 不过遇到多重 \mathcal{A} MS-TeX 重音号时会自动调整位置.

`\doteq` [m] 产生 \doteq .

`\dotfill` 用点线填满一段空间, 例如: `..... = \dotfill`.

`\dots` 产生 \dots .

`\dots` [m][a] 在数学模式里打印 3 个点, 其竖直位置会根据后继符号自动确定.

`\dotsb` [m][a] 接在二元算子后的 `\dots`: \dots .

`\dotsc` [m][a] 接在逗号后的 `\dots`: \dots .

`\dotsi` [m][a] 接在积分号后的 `\dots`: \dots .

`\dotsm` [m][a] 当乘号用的 `\dots`, 同 `\dotsb`: \dots .

`\Downarrow` [m] 产生 \Downarrow .

`\downarrow` [m] 产生 \downarrow .

`\ell` [m] 产生 ℓ .

`\em` 声明选用强调字体, 它与当前字体同族、同系列, 但形状不同, 通常是在直立字体与斜体间转换.

`\emph{文本}` 用强调字体打印文本, 它与当前字体同族、同系列, 但形状不同, 通常是在直立字体与斜体间转换, 此命令等价于 `{\em 文本\em}`, 也就是说, 自动插入了斜体附加留空.

`\emptyset` [m] 产生 \emptyset .

`\encl{附件表}` 文件类 “信件” (letter) 中的命令, 在信末打印 ‘encl.’ 再接上附件表, 词 ‘encl’ 包含在命令 `\enclname` 中, 可根据需要重新定义.

`\enclname` 文件类 “信件” (letter) 中的命令, 含有命令 `\encl` 所打印的词, 在英文环境中是 ‘encl’, 可根据需要重新定义.

`\end{环境名}` 结束由命令 `\begin{环境名}` 进入的环境.

`\enlargethispage{长度}` 使 `\textheight` 暂时增加长度, 以避免出现一个难看的页面, 换页后, `\textheight` 将恢复原值.

`\enlargethispage*{长度}` 类似于 `\enlargethispage`, 不过同时取消所有附加的行间留空, 使这一页能容纳更多内容.

`\ensuremath{数学命令}` 不论在文本模式还是数学模式中使用本命令, 都能使数学命令在数学模式中执行, 这个命令主要用于定义需要数学模式的新命令, 使其在文本模式中也能调用.

`\epsilon [m]` 产生 ϵ .

`\eqref{标记} [a]` 这是 `\ref` 命令的变体, 打印出带括号的由命令 `\label{标记}` 定义的公式编号, 例如: (6.5).

`\equiv [m]` 产生 \equiv .

`\eta [m]` 产生 η .

`\evensidemargin` 偶数页的左页边宽, 仅在文件类“书籍”(book), 或其它文件类但选项 `twoside` 被选取时才有效. 可用 `\setlength` 命令重置新值, 例如:

```
\setlength{\evensidemargin}{2.5cm}
```

`\ExecuteOptions{选项表} [p]` 在一个类文件或宏包里, 本命令将执行选项表里的所有选项的定义, 通常在命令 `\ProcessOptions` 之前调用本命令, 以使某些选项成为默认的.

`\exists [m]` 产生 \exists .

`\exp [m]` 在数学公式中产生函数名‘exp’的命令.

`\extracolsep{附加宽度}` 用于 `tabular` 列格式命令中, 在以后各列之间产生附加宽度的间隔, 本命令用在列格式的 `@` 表达式中:

```
\begin{tabular}{lr@{\extracolsep{2.5mm}}lcr}
```

`\fbox{文本}` 产生一个边框: `\fbox{text} = \boxed{text}`.

`\fboxrule` 由 `\fbox` 或 `\framebox` 产生的边框线条厚度, 可用 `\setlength` 为其重置新值:

```
\setlength{\fboxrule}{1pt}
```

`\fboxsep` 由 `\fbox` 或 `\framebox` 产生的边框与文本的间隔, 可用 `\setlength` 为其重置新值:

```
\setlength{\fboxsep}{1mm}
```

`\fcolorbox 色彩一 色彩二{文本}` 必须先输入宏包 `color` 才能使用本命令, 类似于 `\colorbox`, 文本生成一个带框的 LR 盒子, 以色彩一作为框线颜色, 色彩二作为背景色, 色彩的指定可以都用色彩名, 也可以用同一种模式, 例如:

`\fcolorbox[rgb]{1,0,0}{0,1,0}{Text}`

`\fcolorbox{red}{green}{Text}`

`\figurename` 包含浮动图形标题的命令, 在英文环境里, 将打印出 ‘Figure’, 可被重新定义以适应其它语言.

`\fill` 一个弹性长度, 它的自然长度是0, 但可以无限伸展直至充满整个可用的水平或竖直空间.

`\flat [m]` 产生b.

`\floatpagefraction` 在一个浮动页中必须被浮动单元充满后才能换新页的部分, 这是一个十进小数, 可用以下命令设置新值:

`\renewcommand{\floatpagefraction}{0.9}`

`\floatsep` 两个同处于顶部或底部的浮动单元间的竖直间隔, 可用 `\setlength` 重置新值:

`\setlength{\floatsep}{12pt plus2pt minus4pt}`

`\flushbottom` 增加段落间的留空使得每一页的最后一行处于同样的位置, 对于文件类 “书籍” (book) 以及选项 `twoside`, 这是标准的设置.

`\fnsymbol{计数器}` 把计数器的当前值用 ‘脚注符号’ 打印:

`*†‡§¶||**††††`

`\fontfamily{族}` 本命令选取字体的族, 对标准 L^AT_EX 而言, 族的可能值为: `cmr`, `cmss`, `cmtt`, `cmfi`.

`\fontseries{系列}` 本命令在一个族内选取字体的系列, 对标准 L^AT_EX 而言, 系列的可能值为: `m` (中等粗细), `bx` (粗体).

`\fontshape{形状}` 本命令选取字体的形状, 对标准 L^AT_EX 而言, 形状的可能值为: `n` (正常), `it` (斜体), `sl` (斜体), `sc` (小号大写字母), `u` (直立的意大利体).

`\fontsize{尺寸}{行距}` 本命令选取字体的尺寸, 尺寸是以 `pt` 为单位的整数值, 而行距则给出 `\baselineskip` 的值, 例如:

`\fontsize{12}{14pt}`

`\footnote[数]{文本}` 产生以脚注文本为内容的脚注, 并用可选的数作为编号代替自动编号.

`\footnotemark[数]` 在当前文本产生一个脚注标号, 并用可选的数作为编号代替自动编号, 此命令可用于一些禁用 `\footnote` 的结构里, 如: LR 盒子, 表格, 数学公式等.

`\footnoterule` 这是一个内部命令, 用来在正文与脚注之间画一条水平线, 可被重新定义为:

`\renewcommand{\footnoterule}{\rule{宽度}{高度}\vspace{-高度}}`

`\footnotesep` 在两个脚注间的竖直间距, 可用 `\setlength` 重置新值:

`\setlength{\footnotesep}{6.5pt}`

`\footnotesize` 把字体尺寸转换成 `\footnotesize`, 它比 `\scriptsize` 大, 但比 `\small` 小, .

`\footnotetext[数]{脚注文本}` 产生以脚注文本为内容的脚注, 但不在当前正文中添加脚注标号, 页底脚注前面的标号取自计数器 `footnote` 的当前值, 而且标注后不改变此计数器的值, 当给出可选的数时, 则以此数值作为标号, 本命令可与 `\footnotemark` 配合用于一些禁用 `\footnote` 的结构里, 如: LR 盒子, 表格, 数学公式等, 以插入脚注, 不过 `\footnotetext` 命令必须放在这些结构的外面.

`\footskip` 从正文底部到脚注水平线的下缘间的距离, 可用 `\setlength` 重置新值:

`\setlength{\footskip}{25pt}`

`\forall [m]` 产生 \forall .

`\foreignlanguage{语言}{文本}` 在 `babel` 系统里用指定的语言排印一段短文本.

`\frac{分子}{分母} [m]` 生成分式的数学命令.

`\frame{文本}` 围绕文本产生一个没有间隙的框, 如 `\text{}`, 主要用于 `picture` 环境的 `\put` 或 `\multiput` 命令里.

`\framebox[宽度][位置]{文本}` 产生一个围绕文本的、具有指定宽度的边框, 可选项位置的取值为: `l` 或 `r`, 分别表示文本应在框中居左或居右, 默认值则是居中, 在 L^AT_EX 2_ε 中还可选 `s`, 即把文本伸展到整个宽度.

`\framebox(宽度,高度)[位置]{文本}` `picture` 环境里的绘图元素, 被用在命令 `\put` 或 `\multiput` 的参数中, 它产生一个有指定宽度与高度的方框, 在没有可选参数位置时, 文本放在框的中心, 可选参数位置规定文本在框内的位置: `l`, `r`, `t`, `b` 分别表示左边, 右边, 顶部, 底部, 还可取其中两个字母的组合, 如 `lt` 表示左上角, 在 L^AT_EX 2_ε 中还可选 `s`, 即把文本伸展到整个宽度.

`\frenchspacing` 使用这个命令后, 句点后面不再有额外的留空, 与此相反的命令是 `\nonfrenchspacing`.

`\frontmatter` 在文件类“书籍”(book)中, 本命令用于引入正文前面的内容, 如前言, 目录等, 其作用是关闭命令 `\chapter` 中的章计数功能, 并用罗马数字打印页码.

`\frown [m]` 产生 \curvearrowleft .

`\fussy` 抵消 `\sloppy` 命令的作用, 使得词间距恢复正常.

`\Gamma` [m] 产生 Γ .

`\gamma` [m] 产生 γ .

`\gcd` [m] 在数学公式里产生函数名 ‘gcd’, 本命令允许带下标作为下界.

`\ge` [m] 产生 \geq .

`\genfrac{左定界符}{右定界符}{线厚}{字大小}{上部}{下部}` [m][a] 生成一般分式, 线厚是分式线的厚度, 字大小可取值0-3, 用`\displaystyle`、`\textstyle`、`\scriptstyle`和`\scriptscriptstyle`表示数学字体的大小, 如为空, 则表示自动设置, 例如`\binom`的定义就是:

`\genfrac{()}{0pt}{}{#1}{#2}`

`\geq` [m] 产生 \geq .

`\gets` [m] 产生 \leftarrow .

`\gg` [m] 产生 \gg .

`\glossary{条目}` 若在前言中加入了命令 `\makeglossary`, 则本命令写入一条 `\glossaryentry` 命令到 `glo` 文件, 否则什么也不做.

`\glossaryentry{条目}{页码}` 命令 `\glossary` 在 `glo` 文件中写入的内容.

`\graphpaper[数](x,y)(lx,ly)` 宏包 `graphpap` 为 `picture` 环境增加的命令, 它画出一个带有标注的方格坐标系, 其左下角坐标为 (x,y) , 宽度为 lx , 高度为 ly , 每隔指定数个单位画一条线, 第5条线是粗线, 数的默认值是10, 所有参数都必须为整数.

`\grave{x}` [m] 产生数学变量 x 上的重音号: `\grave{a} = \grave{a}`.

`\Grave{x}` [m] 类似于 `\grave`, 不过遇到多重 $\mathcal{A}\mathcal{M}\mathcal{S}$ -T_EX 重音号时会自动调整位置.

`\guillemotleft` 当 T1 编码被激活时, 产生符号 «.

`\guillemotright` 当 T1 编码被激活时, 产生符号 ».

`\guilsinglleft` 当 T1 编码被激活时, 产生符号 ‹.

`\guilsinglright` 当 T1 编码被激活时, 产生符号 ›.

`\H{x}` 产生重音号: `\H{o} = \ddot{o}`.

`\hat{x}` [m] 产生数学变量 x 上的重音号: `\hat{a} = \hat{a}`.

`\Hat{x}` [m] 类似于 `\hat`, 不过遇到多重 $\mathcal{A}\mathcal{M}\mathcal{S}$ -T_EX 重音号时会自动调整位置.

`\hbar` [m] 产生 \hbar .

`\headheight` 每一页的页眉高度, 可用 `\setlength` 重置新值:

`\setlength{\headheight}{25pt}`

`\headsep` 每一页的页眉底部到正文顶部的竖直距离, 可用 `\setlength` 重置新值:

`\setlength{\headsep}{0.25in}`

`\headtoname` 文件类“信件”(letter)中的命令,在信件的第二页起的页眉上会出现‘To 收信人姓名’,`\headtoname`的内容就是To这个词,为了适用于其它语言,可以对此重新定义.

`\heartsuit` [m] 产生♥.

`\height` 等于一个盒子的自然高度(从基线到顶部的距离)的长度参数,它主要用于命令`\makebox`, `\framebox`或`\savebox`的宽度参数内,或者用于`\parbox`及`minipage`环境的高度参数中,例如:

`\framebox[6\height]{text}`

`\hfill` 水平弹性长度,其自然长度等于0,但可伸展成任意值,它可用于以空白填满一个行,实际上这个命令是`\hspace{\fill}`的缩写.

`\hline` 本命令在`array`与`tabular`环境里产生一条贯穿整个表格宽度的水平线.

`\hoffset` 输出页面与打印机设定的打印起始线之间的水平位移,通常打印起始线与纸边的距离是1in, `\hoffset` 等于0pt,可以用`\setlength`命令重置新值:

`\setlength{\hoffset}{-1in}`

`\hom` [m] 在数学公式里产生函数名‘hom’.

`\hookleftarrow` [m] 产生 \hookleftarrow .

`\hookrightarrow` [m] 产生 \hookrightarrow .

`\hrulefill` 用水平直线填满一段空间,例如: `\hrulefill`.

`\hspace{宽度}` 产生具有指定宽度的水平空间,当它位于行首或行尾时则被忽略.

`\hspace*{宽度}` 产生具有指定宽度的水平空间,即使位于行首或行尾时也不被忽略.

`\huge` 把字体尺寸转换成`\huge`,它比`\Huge`小,比`\LARGE`大.

`\Huge` 把字体尺寸转换成最大的`\Huge`,它比`\huge`更大.

`\hyphenation{表}` [p] 为难拆分的词建立的表,表的形式是:

hy-phen-a-tion per-mit-ted.

`\i` 产生i.

`\idotsint` [m][a] 产生 $\int \cdots \int$.

`\iff` [m] 产生 \iff .

`\IfFileExists{文件名}{是}{否}` 测试在L^AT_EX输入文件的目录里是否存在此文件名的文件,若存在,则执行是提供的命令序列,否则,执行否提供的命令序列,本命令与`\InputIfFileExists`类似,不过这里并不输入.

`\iflanguage{语言}{是}{否}` 在多语言的`babel`系统里,测试语言是否当前选用的语言,若是,则执行是提供的命令序列,否则,执行否提供的命令序列.

`\ifthenelse{测试}{真}{假}` 本命令要在输入了标准宏包`ifthen`后才能使用,如

果逻辑语句测试的执行结果为真, 则执行真提供的命令序列, 否则, 执行假提供的命令序列, 逻辑语句可以是: 关系式(用 $< = >$ 连接的两个数), 奇偶性测试(`\isodd{整数}`), 两个文本的比较(`\equal{文本一}{文本二}`), 两个长度的比较(`\lengthtest{长度一 op 长度二}`, 其中`op`是 $< = >$ 中之一)或布尔开关的测试(`\boolean{开关}`), 这里的布尔开关是由命令`\newboolean{开关}`创立的, 并且由命令`\setboolean{开关}{值}`置值, 其中的值可以为`true`或`false`, 逻辑语句还可以用逻辑算子`\and`, `\or`以及`\not`联合, 并用`\(`与`\)`分组.

测试`\isodd`, `\lengthtest`以及`\boolean`是L^AT_EX 2_ε的新内容.

`\Im [m]` 产生 \Im .

`\imath [m]` 产生 i .

`\in [m]` 产生 \in .

`\include{文件名}` 把名为文件名, 扩展名为`tex`的文件内容插入当前位置, 注意总是另起一个新页! 本命令与`\includeonly`一起, 使得有可能仅处理文件的一部分, 好像其它部分也一起被处理一样.

`\includegraphics[llx, lly][urx, ury]{文件名}` 本命令要输入宏包`graphics`才有效, 它输入文件名指定的文件里的图形, 图形边框的坐标是 (llx, lly) (左下角)以及 (urx, ury) (右上角), 只有这个范围里的图形可接受进一步处理: 缩放或旋转, 文本中也仅为处理后的边框保留空间, 超出这个范围的图形仍被打印出来, 但可能会与相邻文本重叠.

图形边框的坐标可以带有单位, 默认的单位是大点`bp` ($1/72$ in).

如果边框的坐标被忽略, 则可用别的方法得到信息, 取决于图形文件的类型, 对于`eps`文件, 信息可从图形文件本身得到.

如果 llx 与 lly 未被指定, 则取默认值0, 也就是说, 如果值给出一组坐标, 则被认为是右上角坐标.

`\includegraphics*[llx, lly][urx, ury]{文件名}` 类似于`\includegraphics`, 只是边框以外的部分被舍去.

`\includeonly{文件名表}` [p] 只有在文件名表中出现的文件才被`\include`命令读入, 其它文件的`\include`则被忽略, 表中文件名之间用逗号分隔, 而所有的辅助文件都被输入, 使得页码, 章节号以及交叉参考都保持正确.

`\indent` 下一段落的第一行将被缩进.

`\index{条目}` 若前言中加入了命令`\makeindex`, 则本命令写入一条`\indexentry`命令到`idx`文件, 否则什么也不做, 如果这些条目有以下形式:

`\index{主条目}`

`\index{主条目!次条目}`

`\index{主条目!次条目!三级条目}`

则用 MakeIndex 文件处理这个 idx 文件后可以创立一个 theindex 环境, 使得这些条目按字母次序排列, 并用命令 `\item`, `\subitem` 与 `\subsubitem` 加以组织.

`\indexentry{条目}{页码}` 这是 `\index` 命令写在 idx 文件里的条目信息.

`\indexname` 此命令包含索引的标题, 在英文环境里为 'Index', 可根据使用的语言给予重新定义.

`\indexspace` 这是 theindex 环境里的命令, 它产生一个空白行.

`\inf [m]` 在数学公式中产生函数名 'inf' 的命令, 可带下标作为下界.

`\infty [m]` 产生 ∞ .

`\intertext{文本} [m][a]` 本命令在多行数学公式内插入居左对齐的文本行, 而不影响其它公式的对齐.

`\input{文件名}` 把名为文件名, 扩展名为 tex 的文件内容插入当前位置, 在输入文件内可以含有 `\input` 命令.

`\InputIfFileExists{文件名}{是}{否}` 测试在 L^AT_EX 输入文件的目录里是否存在此文件名的文件, 若存在, 则执行是提供的命令序列并且输入此文件, 否则, 执行否提供的命令序列.

`\int [m]` 产生 \int .

`\iint [m][a]` 产生 \iint .

`\iiint [m][a]` 产生 \iiint .

`\iiiiint [m][a]` 产生 \iiiiint .

`\intertextsep` 表示被插入页面中间的浮动单元与上下文本间的竖直间距, 可用命令 `\setlength` 为其重置新值:

`\setlength{\intertextsep}{10pt plus2pt minus3pt}`

`\invisible` 当文件类是“投影片” (slides) 时, 本声明使以下文本用‘隐形墨水’打印, 也就是说, 它不被打印, 但占据空间, 它的作用范围可持续到所在局部环境的结束, 或遇到 `\visible` 命令, 本命令可用于生成覆盖片.

`\iota [m][a]` 产生 ι .

`\it` 转变为罗马族, 斜体形状, 中等粗细系列 (*Roman*, *italic*, *medium*) 的字体属性.

`\itdefault` 本命令定义由 `\itshape` 命令选取的形状属性, 可用 `\renewcommand` 重新定义:

`\renewcommand{\itshape}{it}`

`\item[标签]` 在列表 (list) 环境中生成一个标签, 并引出条目的文本, 当没有可选项时, 标签是根据环境的类型生成的, 例如 `enumerate` 环境中是数, 否则, 用选项标签取代.

`\item{条目}` 在 `theindex` 环境里生成一个主条目.

`\itemindent` 在 `list` 环境里, 标签以及每个条目的第一行文本的缩进量, 其标准值为 0pt, 但可用 `\setlength` 为其重置新值:

`\setlength{\itemindent}{1em}`

`\itemsep` 在 `list` 环境里, 不同条目的间距等于 `\parsep` 加上 `\itemsep`, 可用命令 `\setlength` 为其重置新值:

`\setlength{\itemsep}{2pt plus1pt minus1pt}`

`\itshape` 本声明把字体的形状属性改为斜体, 但保留族与系列不变.

`\j` 产生 j .

`\jmath [m]` 产生 j .

`\Join [m]` 产生 \Join .

`\jot` 在 `eqnarray` 或 `eqnarray*` 环境里, 公式行之间的竖直距离, 其标准值等于 3pt, 可用 `\setlength` 为其重置新值:

`\setlength{\jot}{4.5pt}`

`\k{x}` 当 T1 编码被激活时, 产生重音号: `\k{A} = \grave{A}`.

`\kappa [m]` 产生 κ .

`\ker [m]` 在数学公式中产生函数名 ‘ker’ 的命令.

`\kill` 在 `tabbing` 环境里, 放在样本行的末尾, 使得样本行只用于设定指标位的位置, 而不被打印出来.

`\L` 产生 L .

`\l` 产生 l .

`\label{标志}` 在当前位置设立一个具有指定名称的标志, 以供文件中其它地方 (可以在它的前面) 用命令 `\ref{标志}` 引用, 打印出章节号、方程号或图号等, 也可用命令 `\pageref{标志}` 引用, 打印页号.

`\labelnumn` 用在 `enumerate` 环境里, 这是一组命令, 可生成嵌套层次的标准标签, n 的取值为 i, ii, iii, iv , 例如: 命令

`\renewcommand{\labelnumii}{\arabic{enumii}.)}`

把 `enumerate` 环境的第二级标准标签修改成 1.), 2.), 等等.

`\labelitemn` 用在 `itemize` 环境里, 这是一组命令, 可生成嵌套层次的标准标签, n 的取值为 i, ii, iii, iv , 例如: 命令

`\renewcommand{\labelitemi}{\(\rightarrow\)}`

把 `itemize` 环境的最外层标准标签修改成 \Rightarrow .

`\labelsep` 在 `list` 环境里, 标签与文本间的距离, 可用 `\setlength` 为其重置新值:

`\setlength{\labelsep}{5pt}`

`\labelwidth` 在 `list` 环境里, 放标签的盒子的宽度, 可用 `\setlength` 为其重置新值:

`\setlength{\labelwidth}{2.2cm}`

`\Lambda` [m] 产生 Λ .

`\lambda` [m] 产生 λ .

`\langle` [m] 产生 \langle .

`\language{数}` 在 T_EX 3.0 版以后, 不同语言的拆分词模式用不同数字作标志, 在执行 `initex` 时, 必须先给出命令 `\language{数}`, 让程序选取适当的分词模式输入格式文件.

`\large` 把字体尺寸转换成 `\large`, 它比 `\Large` 小, 比 `\normalsize` 大.

`\Large` 把字体尺寸转换成 `\Large`, 它比 `\LARGE` 小, 比 `\large` 大.

`\LARGE` 把字体尺寸转换成 `\LARGE`, 它比 `\huge` 小, 比 `\Large` 大.

`\LaTeX` 产生 L^AT_EX.

`\LaTeXe` 产生 L^AT_EX 2_ε.

`\lceil` [m] 产生 \lceil .

`\ldots` [m] 产生 \dots .

`\le` [m] 产生 \leq .

`\leadsto` [m] 产生 \leadsto .

`\left` 定界符 [m] 调整定界符的大小使其适应夹在 `\left ... \right` 内的公式的高度, 例如: `\left[`, 如果另一端没有明显的定界符与之配对, 则可使用空定界符 (`\right.`).

`\Leftarrow` [m] 产生 \Leftarrow .

`\leftarrow` [m] 产生 \leftarrow .

`\leftharpoondown` [m] 产生 \leftharpoondown .

`\leftharpoonup` [m] 产生 \leftharpoonup .

`\lefteqn{公式}` [m] 环境 `eqnarray` 的内部命令. 此命令虽然打印出公式, 但又把它的宽度当成 0, 使它不影响列的宽度, 这个命令主要用于多行公式的第一行.

`\leftmargin` 在 `list` 环境里, 这是环境内文本的左边界与外部正文的左边界相比的缩进值, 可用命令 `\setlength` 为其重置新值. 此外, 在多层次的 `list` 环

境里, 可以在此命令的后面添加 i..iv 表示各层次的缩进值, 例如:

`\setlength{\leftmarginiii}{0.5cm}`

`\Leftrightarrow [m]` 产生 \Leftrightarrow .

`\leftrightharrow [m]` 产生 \leftrightharrow .

`\leftroot{数} [m][a]` 使 `\sqrt` 的根指数稍微左移一点, 例如:

`\sqrt{\leftroot{-1}\uproot{3}\beta}{k}`

`\leq [m]` 产生 \leq .

`\lfloor [m]` 产生 \lfloor .

`\lg [m]` 在数学公式中产生函数名 'lg' 的命令.

`\lhd [m]` 产生 \lhd .

`\lim [m]` 在数学公式中产生函数名 'lim' 的命令, 可带下标作为下界.

`\liminf [m]` 在数学公式中产生函数名 'liminf' 的命令, 可带下标作为下界.

`\limits [m]` 把上下界放在相应符号的上方与下方, 而正常情况是放在后方的.

`\limsup [m]` 在数学公式中产生函数名 'limsup' 的命令, 可带下标作为下界.

`\line($\Delta x, \Delta y$){长度}` 本命令是 `picture` 环境里的一个图形元素, 通常作为 `\put` 或 `\multiput` 的参数, 其作用是绘制水平或竖直直线段以及某些特定斜率的斜线段. 对于水平或竖直直线段, 长度就是线段的实际长度 (以 `\unitlength` 为单位), 对于斜线段, 长度是它在 x 轴上的投影 (即水平位移). ($\Delta x, \Delta y$) 给出了线段的斜率, 其中 Δx 与 Δy 是一对互素整数, 满足 $-6 \leq \Delta x \leq 6, -6 \leq \Delta y \leq 6$.

`\linebreak[n]` 鼓励在此处换行, 并使此行在左右两端都对齐, 换行的迫切性由整数 n 的值确定, $0 \leq n \leq 4$, 4 表示强迫换行.

`\linethickness{厚度}` 在 `picture` 环境里设置水平或竖直直线段的厚度, 厚度是带有单位的长度, 如: 1.2mm.

`\listfigurename` 此命令含有插图目录的标题, 在英语中定义为 'List of Figures', 但可被重新定义以适用于其他语言.

`\listfiles [p]` 如果在前言里给出了此命令, 则在处理过程中所有读入的文件都会在程序结束后显示在监视器上, 同时也被写入纪录文件. 文件列表中包括版本号, 日期以及其它附加信息.

`\listoffigures` 产生一个插图目录, 其条目内容取自 `figure` 环境中的 `\caption` 命令.

`\listoftables` 产生一个表格目录, 其条目内容取自 `table` 环境中的 `\caption` 命令.

`\listparindent` 在 `list` 环境里, 段落的第一行的缩进量, 可用 `\setlength` 命令

重置新值:

`\setlength{\listparindent}{1em}`

`\listtablename` 此命令含有表格目录的标题, 在英语中定义为 ‘List of Tables’, 但可被重新定义以适用于其他语言.

`\ll` [m] 产生 \ll .

`\ln` [m] 在数学公式中产生函数名 ‘ln’ 的命令.

`\LoadClass` [选项]{类}[版本号] [p] 本命令只能出现在类文件里, 以输入另一个类文件, 在一个类文件里只能使用一次, 被输入文件的扩展名必须是 `cls`, 而且在 `\documentclass` 命令中出现的选项不作为整体选项传递.

选项版本号的格式是: `yyyy/mm/dd`, 例如: 1994/08/01. 如果类文件的日期比它早, 就会打印一个警告信息.

`\location`{数} 当文件类是 “信件” (`letter`) 时, 输入发信人的房间号码. 在标准 L^AT_EX `letter` 类中, 仅当 `\address` 没有出现时才输出这个数, 其目的是用于公司的信头.

`\log` [m] 在数学公式中产生函数名 ‘log’ 的命令.

`\Longleftarrow` [m] 产生 \Longleftarrow .

`\longleftarrow` [m] 产生 \longleftarrow .

`\Longleftrightarrow` [m] 产生 \Longleftrightarrow .

`\longleftrightarrow` [m] 产生 \longleftrightarrow .

`\longmapsto` [m] 产生 \longmapsto .

`\Longrightarrow` [m] 产生 \Longrightarrow .

`\longrightarrow` [m] 产生 \longrightarrow .

`\lq` 产生 ‘, 与 ‘ 符号完全相同.

`\lvert` [m][a] 产生 | (左定界符).

`\lVert` [m][a] 产生 || (左定界符).

`\mainmatter` 在文件类 “书籍” (`book`) 中, 本命令宣告进入书籍的正文, 它把页码计数器重置为 1, 且用阿拉伯数字打印, 恢复 `\chapter` 命令中的章节计数功能, 总之, 消除 `\frontmatter` 命令的后果.

`\makebox` [宽度][位置]{文本} 产生一个有指定宽度且包含文本的盒子, 选项位置决定了文本在盒子中的位置, `l` 或 `r` 分别表示居左或居右, 默认是居中, L^AT_EX 2_ε 中增加了一个 `s`, 表示伸展到整个宽度.

`\makebox`(宽度, 高度)[位置]{文本} 这是 `picture` 环境里的一个绘图命令, 用作 `\put` 或 `\multiput` 命令的参数, 它产生一个有指定宽度与高度且包含文本的盒子. 选项位置是指文本在盒子内部的位置, 它们可以是居左 (`l`), 居右 (`r`),

居顶(t)或居底(b),也可以是两者的组合,如lt,默认值是居中. L^AT_EX 2_ε 中增加了一个s,表示伸展到整个宽度.

`\makeglossary [p]` 激活正文中的 `\glossary` 命令.

`\makeindex [p]` 激活正文中的 `\index` 命令.

`\makeindex{文本}` 被命令 `\item` 调用的内部命令,它在list环境里产生一个内容为文本的标签.

`\makelabels` 当文件类是“信件”(letter)时,利用 `\begin{letter}` 环境里的条目产生地址标签.

`\MakeShortVerb{\字母}` 当输入了标准宏包 `shortvrb` 后,本命令使字母成为 `\verb` 字母的缩写,于是夹在两个字母中间的文本都被按照字符的本来意义打印.例如在发出命令 `\MakeShortVerb{\|}` 后,输入 `| \cmd{} |` 等价于输入 `\verb| \cmd{} |`. 其打印输出为 `\cmd{}`. 在发出命令 `\DeleteShortVerb{\|}` 后,字符|又恢复其本来意义.

`\maketitle` 利用 `\author`, `\title` 以及 `\date`, `\thanks` 中包含的信息产生一个标题页.

`\mapsto [m]` 产生 \mapsto .

`\marginpar[左文本]{右文本}` 在页面的右边空白处产生内容为右文本的注记. 在双页格式中遇到偶数页时,页边注记打印在左方,这时就用可选项左文本替代. 在双栏格式里,页边注记总是打印在外侧,这时也用左文本代替左页边的注记.

`\maginparpush` 两个页边注记间的最小竖直距离,可用 `\setlength` 重置新值.

`\maginparsep` 正文与页边注记间距离,可用 `\setlength` 重置新值.

`\maginparwidth` 页边注记所在盒子的宽度,可用 `\setlength` 重置新值.

`\markboth{左边}{右边}` 在双页格式里,如果把页面格式选为 `myheadings`,或当页面格式 `headings` 的自动页眉设置被改变后,本命令设置左右页眉的文本.

`\markright{左边}{右边}` 如果把页面格式选为 `myheadings`,或当格式 `headings` 的自动页眉设置被改变后,本命令设置页眉的文本.如果是双页格式,则只设置右页眉.

`\mathbf{文本} [m]` 在数学模式里用粗体(`\bfseries`)打印文本,其中的空格被忽略.

`\mathcal{文本} [m]` 在数学模式里用书写体打印文本,本命令取代了 L^AT_EX 2.09 中的 `\cal`.

`\mathindent` 当选项 `fleqn` 被选中后,行间公式在左边的缩进量,可用 `\setlength` 为其重置新值:

`\setlength{\mathindent}{25pt}`

`\mathit{文本}` [m] 在数学模式里用斜体(`\itshape`)打印文本, 它与`\mathnormal`的区别可从下例看出: 比较 *mathit* 与 *mathnormal*.

`\mathnormal{文本}` [m] 在数学模式里用数学字体打印文本, 它取代了 L^AT_EX 2.09 中的 `\mit`.

`\mathring{x}` [m] 产生数学模式里的重音号: `\mathring{a} = \mathring{a}`.

`\mathrm{文本}` [m] 在数学模式里用罗马体(`\rmfamily`)打印文本, 其中的空格被忽略.

`\mathsf{文本}` [m] 在数学模式里用无衬线字体(`\sffamily`)打印文本, 其中的空格被忽略.

`\mathtt{文本}` [m] 在数学模式里用打字机字体(`\ttfamily`)打印文本, 其中的空格被忽略.

`\mathversion{变体}` 选取当前的数学变体, 可能的值为 `bold` 与 `normal`, 用命令 `\DeclareMathVersion` 可创立新的变体.

`\max` [m] 在数学公式中产生函数名 ‘max’ 的命令, 可带下标作为下界.

`\mbox{文本}` 围绕文本建立一个 LR 盒子.

`\mddefault` 定义了由命令 `\mdseries` 选取的字体系列, 可用 `\renewcommand` 予以重定义:

`\renewcommand{\mddefault}{m}`

`\mdseries` 本声明不改变当前字体的族与形状, 但转变成中等粗细 medium 序列.

`\medskip` 插入值为 `\medskipamount` 的竖直间隔, 参见 `\bigskip`, `\smallskip`.

`\medskipamount` 用于 `\medskip` 的竖直间隔的标准值, 可用命令 `\setlength` 加以改变:

`\setlength{\medskipamount}{3ex plus 1ex minus 1ex}`

`\medspace` [m][a] 这是 `\:` 的别名, 用在数学公式中插入中等大小的空白.

`\MessageBreak` 在出错信息、警告信息或版本注解信息的文本中产生强迫换行.

`\mho` [m] 产生 \mathcal{U} .

`\mid` [m] 产生 $|$.

`\min` [m] 在数学公式中产生函数名 ‘min’ 的命令, 可带下标作为下界.

`\mit` 这是 L^AT_EX 2.09 的命令, 表示选取数学斜体, 在 L^AT_EX 2_ε 中已被 `\mathnormal` 取代.

`\mod{变量}` [m][a] 在输入宏包 `amsopn` 与 `amsmath` 后, 本命令数学公式中产生函数名 ‘mod’, 例如:

`y\mod{a+b} = y \mod a + b`

`\models` [m] 产生 \models .

`\mp` [m] 产生 \mp .

`\mspace{长度}` [m][a] 在数学公式里插入指定长度的空白, 长度单位是 μ ($= 1/18\text{em}$), 例如: `\mspace{-4\mu}`

`\mu` [m] 产生 μ .

`\multicolumn{列数}{列格式}{文本}` 在 `tabular` 和 `array` 环境里把指定个数的列组合成单个列, 列格式指定组合后单个列的格式, 可取值 `l`, `c`, `r` (只能一个) 以及添加竖线 `|`.

`\multipt(x,y)(\Delta x,\Delta y){个数}{图形元素}` 在 `picture` 环境里以 (x,y) 作为图形元素的基准点坐标, 以 $(\Delta x,\Delta y)$ 作为平移向量, 画出指定个数的图形元素.

`\multlinegap` [m][a] 在 $\mathcal{A}\mathcal{M}\mathcal{S}$ -L^AT_EX 的 `multline` 环境里产生的多行公式的左右缩进量, 初始值是 `10pt`, 用户可重置新值.

`\nabla` [m] 产生 ∇ .

`\name{发信人}` 在文件类 “信件” (`letter`) 中用以输入发信人姓名.

`\natural` [m] 产生 \natural .

`\nearrow` [m] 产生 \nearrow .

`\NeedsTeXFormat{格式}[版本]` [p] 声明为了处理这个文件所需的 T_EX 格式, 这应该是文件的第一个语句. 到目前为止, 格式的合法值仅有 `LaTeX2e`. 可选项版本应该输入一个格式为 `yyyy/mm/dd` 的日期, 指明与本文件兼容的最早的版本日期, 例如:

`\NeedsTeXFormat{LaTeX2e}[1994/06/01]`

`\neg` [m] 产生 \neg .

`\negmedspace` [m][a] 在数学公式中插入一个负的中等大小的空白.

`\negthickspace` [m][a] 在数学公式中插入一个负的较大的空白.

`\negthinspace` [m][a] 这是 `\!` 的别名, 在数学公式中插入一个负的小间隙.

`\newboolean{开关}` 使用本命令必须先输入标准 L^AT_EX 宏包 `ifthen`, 它创建一个新的布尔开关, 开关的值由命令 `\setboolean{开关}{值}` 指定, 这里的值可取 `true` 或 `false`. 检测开关的值的命令是 `\boolean{开关}`, 这个检测命令可用于 `\ifthenelse` 或 `\whiledo` 的测试部分.

`\neq` [m] 产生 \neq .

`\newcommand{\命令名}[参量个数][选项]{命令内容}` 定义用户命令 `\命令名` 使其具有指定的命令内容, 选项参量个数 ≤ 9 , 命令内容中用 `#1, \dots, \#参量个数` 代表这些参量. 如果本命令还带有第二个选项 (L^AT_EX 2.09 无此功能), 则说明新命令的第一个参量是可选的, 而且当它不被给出时以这里的选项作为默认

值.

`\newcommand*`{\命令名}[参量个数][选项]{命令内容} 类似于 `\newcommand`, 不过命令内容必须是短的, 即在命令内容中不能出现新的段落, 也就是说, 不能有 `\par` 及空白行.

`\newcounter`{计数器名}[主计数器名] 建立一个具有指定计数器名的新计数器. 可选参数主计数器名是一个已有的计数器, 使得每当主计数器的值变化时, 新建计数器被置零, 也就是说, 使新计数器成为从计数器.

`\newenvironment`{环境名}[参量个数][选项]{进入命令}{退出命令} 定义有指定环境名的用户环境, 当遇到 `\begin` 命令进入此环境时, 就执行进入命令, 遇到 `\end` 命令退出时, 则执行退出命令. 选项参量个数 ≤ 9 , 在进入命令中用 `#1, ..., #参量个数` 代表这些参量. 如果本命令还带有第二个选项 (L^AT_EX 2.09 无此功能), 则说明 `\begin` 命令的第一个参量是可选的, 而且当它不被给出时以这里的选项作为默认值.

`\newenvironment*`{环境名}[参量个数][选项]{进入命令}{退出命令} 类似于 `\newenvironment`, 不过进入命令必须是短的, 不能出现新的段落, 也就是说, 不能有 `\par` 及空白行.

`\newfont`{\字体命令}{字库文件 scaled 放大倍数} 在 \字体命令与外部的 (经过适当放缩的) 字库文件之间建立一个关联, 执行这个 \字体命令后, 基线间隔 `\baselineskip` 以及行距等都保持原样.

`\newlength`{\长度命令} 创建一个名为 \长度命令的新命令, 其初始值为 0pt, 可用 `\setlength` 为其重置新值, 例如:

```
\setlength{\lengthcmd}{1mm}
```

其中的长度必须带有单位, 也可以是弹性长度.

`\newline` 结束当前行, 但不向右对齐.

`\newpage` 结束当前页, 剩下部分保持空白.

`\newsavebox`{\盒子名} 创建一个名为 \盒子名的新的存储盒子, 可用 `\savebox` 命令把 LR 盒子保存在其中.

`\newtheorem`{定理环境名}[编号]{标题}

`\newtheorem`{定理环境名}{标题}[主计数器名] 定义一个名为定理环境名的新的定理环境, 每当这个环境被调用时, 先用黑体字打印标题 (例如 **Theorem**, **Proposition** 等), 再打印一个自动生成的序号, 然后用斜体打印环境内部的文本. 选项编号是另一个定理环境的名字, 它与当前的环境共用同一个序号计数器. 另一个可选项主计数器名是章节计数器的名称, 例如 `chapter` 或 `section`, 每当主计数器值改变时, 就重置定理计数器的值, 使得定理计数器

成为从计数器. 两个可选项每次只能选一个.

`\NG` 当 T1 编码被激活时, 产生字母 l).

`\ng` 当 T1 编码被激活时, 产生字母 n).

`\ni` [m] 产生 \ni .

`\nocite{引用词}` 参考文献数据库里具有引用词的条目将被包含在文献目录里, 即使在文本中未被引用. 如果用 `\nocite{*}`, 则数据库里的所有条目都将被包含在文献目录里.

`\nocorr` 与字体命令 `\emph`, `\textsl`, `\textit` 合用. 以取消自动的斜体校正, 如:

`\nocorr\emph{emphatic text}` 或

`\emph{emphatic text\nocorr}`

`\nofiles` [p] 不输出辅助文件 `aux`, `glo`, `idx`, `lof`, `lot` 以及 `toc`.

`\noindent` 使下一段落的第一行不缩进.

`\nolimits` [m] 把上下界放在符号的后面, 而不是放在上下.

`\nolinebreak[n]` 建议此处不要换行, 其迫切程度由整数 n 确定, $0 \leq n \leq 4$, n 取 4 时相当于没有选项, 意为禁止换行.

`\nonfrenchspacing` 与 `\frenchspacing` 相反, 使用这个命令后, 句子后面加上额外的留空.

`\nonumber` [m] 在 `eqnarray` 环境的多行公式里, 有此命令的行将没有方程编号.

`\nopagebreak[n]` 建议不要在此处换页, n 取 0 到 4 的整数, 迫切性随 n 的增加而增大. $n = 4$ 时相当于不含可选项的命令, 禁止换页.

`\normalcolor` 如果输入了宏包 `color`, 本命令把文字的颜色重置为前言末尾的颜色, 通常是黑色. 前言中的 `\color` 命令可以改变‘标准’色彩. 本命令主要用于 L^AT_EX 内部的宏包中, 使得在打印标题后可以重置文本的颜色, 其它与 `color` 相容的宏包也能使用它.

`\normalfont` 本声明把字体转换为默认的族、形状和系列.

`\normalmarginpar` 取消命令 `\reversemarginpar` 的作用, 使页边注记放在标准位置, 即右边或外侧.

`\normalsize` 本命令把字体尺寸转换成 `\normalsize`, 也就是 `\documentclass` 或 `\documentstyle` 命令的选项中选取的大小, 它比 `\large` 小, 比 `\small` 大.

`\not` [m] 使后面的比较符号取否定的意义: `\not\cong` = $\not\cong$.

`\notag{标记}` [m][a] 用在 $\text{\textit{AMS-LAT}}\text{\textit{E}}\text{\textit{X}}$ 的对齐环境中. 取消公式自动编号.

`\notesname` 含有注记命令 `\notes` 的标题的命令, 在英语中定义为‘notes’, 但可被重新定义以适用于其他语言.

`\notin` [m] 产生 \notin .

`\nu [m]` 产生 ν .

`\numberwithin{计数器}{主计数器} [a]` 把计数器重新定义为另一主计数器的子计数器, 也就是说, 每当主计数器的值增加时, 此计数器就置零. 打印计数器的值时一起打印主计数器的值, 这条命令通常用来使论文中的方程能与节一起计数, 如:

`\numberwithin{equation}{section}`

`\O` 产生 \mathcal{O} .

`\o` 产生 \mathcal{o} .

`\oddsidemargin` 在文件类“书籍”(book)中, 或其它文件类但选项 `twoside` 被选取时, 本命令设置奇数页的左页边宽, 而在其余情形, 本命令设置所有页的左页边宽. 可用 `\setlength` 命令重置新值, 例如:

`\setlength{\oddsidemargin}{1.5cm}`

`\odot [m]` 产生 \odot .

`\OE` 产生 \mathbb{O} .

`\oe` 产生 \mathbb{o} .

`\oint [m]` 产生 \oint .

`\Omega [m]` 产生 Ω .

`\omega [m]` 产生 ω .

`\ominus [m]` 产生 \ominus .

`\onecolumn` 开始一个新页, 并从双栏格式转换成单栏格式.

`\onlynotes{数表}` 在文件类“投影片”(slides)中, 本命令出现在前言中, 使得只产生数表中出现的数字对应的注记, 本命令的作用与 `\onlyslides` 类似.

`\onlyslides{数表}` 在文件类“投影片”(slides)中, 本命令出现在前言中, 使得只产生数表中出现的数字对应的投影片. 数表中的数以逗号分隔, 且可含有以连字符表示的范围, 如:

`\onlyslides{4,10-13,23}`

`\opening{称呼语}` 在“信件”(letter)类的 letter 环境里, 它设置了信件开头的称呼语, 例如: `\opening{Dear George,}`.

`\oplus [m]` 产生 \oplus .

`\OptionNotUsed [p]` 本命令只能用于选项的定义中, 特别是默认选项的定义中, 它声明 `\CurrentOption` 是未经处理的. 本命令用于这样的情况: 由于某些原因, 例如缺少必要的文件, 默认的选项可能会失败, 这时本命令就通知 L^AT_EX, 所需的选项尚有待处理.

`\oslash [m]` 产生 \oslash .

`\otimes` [m] 产生 \otimes .

`\oval(宽度,高度)[部分]` 本命令是 `picture` 环境里的图形元素, 产生一个具有指定宽度与高度的圆角矩形, 可选参数部分可取值 `t`, `b`, `l`, `r`, 分别表示只画上半部, 下半部, 左半部或右半部, 如果取其中两个的组合, 例如 `tl` 或 `lt`, 则表示画左上角的四分之一部分. 本命令一般用作 `\put` 或 `\multiput` 命令的参数.

`\overbrace{公式}` [m] 在公式的上面产生一个水平的花括号, 以上标形式出现在本命令后面的数学符号将被居中放在花括号的上方. 例如:

$$\overbrace{a+b} = a + b$$

$$\overbrace{x+y+z}^{\xi\eta\zeta} = x + y + z$$

`\overleftarrow{公式}` [m][a] 在公式的上面画一条指向左方的长箭头.

`\overleftrightarrow{公式}` [m][a] 在公式的上面画一条左右两边都有箭头的水平线.

`\overline{公式}` [m] 在公式的上面产生一条水平直线:

$$\overline{a-b} = a - b$$

`\overrightarrow{公式}` [m][a] 在公式的上面画一条指向右方的长箭头.

`\overset{字符}{\符号}` [m][a] 把字符按上标的字体放在 `\符号` 上方:

$$\overset{\alpha}{\longrightarrow}$$

`\P` 产生 ¶.

`\PackageError{宏包名}{出错信息}{帮助}` [p] 本命令只能出现在宏包文件中, 它向监视器以及纪录文件发出以宏包名为标号的出错信息, 中断进程并等待用户的反应, 如果用户键入 `H<回车>`, 就把帮助打印出来, 在出错信息和帮助中都可使用 `\MessageBreak` 命令换行, 用 `\space` 强制产生空格, 并可用 `\protect` 冠在一个命令前面, 使得这个命令不被解释执行而是把它的名字打印出来.

`\PackageInfo{宏包名}{信息}` [p] 类似于 `\PackageWarningNoLine`, 不过信息不在监视器上显示, 只是写入纪录文件.

`\PackageWarning{宏包名}{警告信息}` [p] 本命令只能出现在宏包文件中, 它向监视器以及纪录文件发出以宏包名为标号的警告信息, 同时给出输入文件的当前行号, 并继续进行下去, 警告信息可按 `\PackageError` 同样的方法格式化.

`\PackageWarningNoLine{宏包名}{警告信息}` [p] 类似于 `\PackageWarning`, 只是不打印当前行号.

`\pagebreak[n]` 鼓励在此处换页, 换页的迫切性由整数 n 的值确定, $0 \leq n \leq 4$, 4 表示强迫换页, 相当于不含选项的命令.

`\pagecolor` 色彩 必须先输入宏包 `color` 才能使用本命令, 它设定从本页开始的

背景色, 以后的页都使用这种背景色, 直至再遇到命令 `\pagecolor` 为止. 色彩的定义可参看 `\color`.

`\pagename` 本命令用于文件类“信件”(letter)中, 用在从第二页起冠在页码前面. 在英文环境中是‘Page’, 可根据使用的语言加以改变.

`\pagenumbering{格式}` 确定页码的格式, 并把页计数器置 1. 格式的可能值是: arabic, roman, Roman, alph, Alph.

`\pageref{标记}` 打印标记所在处的页码, 标记由命令 `\label{标记}` 定义.

`\pagestyle{格式}` [p] 确定页格式, 也就是每页的页眉与页脚的内容, 格式的可能值为: plain, empty, headings, myheadings.

`\paperheight` 由 `\documentclass` 命令中的页面大小选项所确定的页面的总高度, 在默认的 `lettersize` 选项下, 这是 11in; 若选 `a4paper`, 则为 29.7cm. 选项 `landscape` 交换 `\paperwidth` 与 `\paperheight` 的值.

`\paperwidth` 由 `\documentclass` 命令中的页面大小选项所确定的页面宽度, 在默认选项 `lettersize` 下, 这是 8.5in; 若选 `a4paper`, 则为 21cm. 选项 `landscape` 交换 `\paperwidth` 与 `\paperheight` 的值.

`\par` 结束当前段落, 开始一个新的段落. 本命令相当于一个空白行.

`\paragraph[短标题]{标题}` 章节层次中的倒数第二级, 介于 `\subsubsection` 与 `\subparagraph` 之间, 它在当前小节编号数的后面添加自动生成的段落序号, 后面再打印标题. 可选项短标题用在目录中代替标题.

`\paragraph*{标题}` 类似于 `\paragraph`, 只是没有编号, 也不在目录中出现.

`\parallel [m]` 产生 \parallel .

`\parbox[位置][高度][内部位置]{宽度}{文本}` 产生具有指定宽度的竖直盒子, 其中的文本保持左右对齐, 这个盒子相对于周围环境的竖直位置由可选项位置确定: t 表示与顶上的行对齐, b 表示与底下的行对齐, 默认是与中间对齐. 另两个可选项 (L^AT_EX 2.09 无此功能): 高度给出盒子的总高度, 内部位置给出文本在盒子内部的竖直位置, 可能的值为: t 表示居顶, b 表示居底, c 表示居中, s 伸展到整个高度, 其默认值取外部位置指定的值. 高度中可包含下列参数: `\height`, `\depth`, `\width` 与 `\totalheight`.

`\parindent` 段落第一行的缩进量, 可用 `\setlength` 为其指定新值:

`\setlength{\parindent}{1.5em}`

`\parsep` 在 list 环境里段落间的竖直间隔, 可用 `\setlength` 为其指定新值:

`\setlength{\parsep}{2pt plus1pt minus1pt}`

`\parskip` 段落间的竖直距离, 可用 `\setlength` 为其重置新值:

`\setlength{\parskip}{3pt plus1pt minus2pt}`

`\part[短标题]{标题}` 章节层次中的最高级, 它开始新的‘篇’(Part), 自动生成一个编号, 后接标题, 后面的节号不受篇号的影响. 如果给出了选项短标题, 则它在目录里将取代标题.

`\part*{标题}` 类似于 `\part`, 不过它没有编号, 也不出现在目录中.

`\partial [m]` 产生 ∂ .

`\partname` 此命令包含篇的标题, 在英文环境里是‘Part’, 可根据不同语言加以修改.

`\partopsep` 在 `list` 环境里, 如果有一个空白行出现在此环境的前面或后面, 本命令给出了附加的竖直空间. 可用 `\setlength` 命令重置新值:

`\setlength{\partopsep}{2pt plus1pt minus1pt}`

`\PassOptionsToClass{选项}{类} [p]` 把选项传送到后面要用 `\LoadClass` 命令输入类文件中, 本命令必须出现在类文件或被类文件读入的文件中, 它可被用在选项的定义中, 或用在配置文件中, 以激活某个选项.

`\PassOptionsToPackage{选项}{宏包} [p]` 把选项传送到后面用 `\RequirePackage` 命令输入的宏包文件中, 本命令可出现在类文件或宏包文件中, 它被用在选项的定义中, 或用在配置文件中, 以激活某个选项.

`\perp [m]` 产生 \perp .

`\Phi [m]` 产生 Φ .

`\phi [m]` 产生 ϕ .

`\Pi [m]` 产生 Π .

`\pi [m]` 产生 π .

`\pm [m]` 产生 \pm .

`\pmb{符号} [m][a]` 在输入了宏包 `amsmath` 或 `amsbsy` 后, 本命令用多次重复打印的方法把符号打印成黑体.

`\pmod{变量} [m]` 在数学公式里产生带括号的函数名‘mod’:

$y \pmod{a+b} = y \pmod{a+b}$

`\pod{变量} [m][a]` 在输入了宏包 `amsmath` 或 `amsopn` 后, 本命令类似于 `\pmod`, 但不出现函数名‘mod’:

$y \pod{a+b} = y (a+b)$

`\poptabs` 在 `tabbing` 环境里, 恢复前面用 `\pushtabs` 命令存储起来的制表位.

`\pounds` 产生 \pounds .

`\Pr [m]` 在数学公式里产生函数名‘Pr’, 而且可带下标作为下界.

`\prec [m]` 产生 \prec .

`\preceq [m]` 产生 \preceq .

`\prefacename` 本命令包含由命令 `\preface` 产生的标题, 在英文环境里是 ‘Preface’, 可根据不同的语言予以重定义.

`\prime [m]` 产生 \prime (同字符 ‘).

`\printindex` 文件 `makeidx.sty` 里定义的命令, 在用程序 `Makeindex` 处理了 `idx` 文件后, 本命令可生成 `theindex` 环境.

`\ProcessOptions [p]` 在类文件或宏包文件里, 本命令通过执行 `\ds@` 命令来处理被选中的选项, 按照这些命令被定义的次序执行, 然后这些 `\ds@` 被删除.

`\ProcessOptions* [p]` 类似于 `\ProcessOptions`, 不过它是按照选项被选中的次序执行 `\ds@` 命令, 这相当于 L^AT_EX 2.09 的命令 `\@options`, 为了保持兼容性, 这条命令仍然有效.

`\prod [m]` 产生 \prod .

`\propto [m]` 产生 \propto .

`\protect` 当一条脆弱命令被冠以 `\protect` 后, 就能被用于移动命令中, 例如:

```
\section{The \protect\pounds{} Sign}
```

`\providecommand{\命令名}[参量个数][选项]{命令内容}` 类似于 `\newcommand`, 不过当 `\命令名` 已经存在时, 命令内容被忽略.

`\providecommand*{\命令名}[参量个数][选项]{命令内容}` 类似于不带星号的命令 `\providecommand`, 不过命令内容必须是短的, 即在命令内容中不能出现新的段落, 也就是说, 不能有 `\par` 及空白行.

`\ProvidesClass{类}[版本] [p]` 本命令出现在类文件的头部, 声明类名及其版本, 供输入类文件的命令 `\documentclass` 或 `\LoadClass` 核对. 选项版本分成三个部分: 日期, 版本号以及其它信息, 例如:

```
\ProvidesClass{thesis}[1995/01/25 v2.8 U of City]
```

`\ProvidesFile{文件名}[版本]` 本命令出现在一般文件的头部, 声明其名字及版本. 当用命令 `\input` 输入此文件时, 不进行任何检查, 但当 `\listfiles` 被激活时, 这些信息会被显示出来. 这个命令与其他 `\Provides...` 命令不同, 不必限制在前言内.

`\ProvidesPackage{类}[版本] [p]` 本命令出现在宏包文件的头部, 声明宏包名及其版本, 供输入宏包文件的命令 `\usepackage` 或 `\RequirePackage` 核对. 选项版本分成三个部分: 日期, 版本号以及其它信息, 例如:

```
\ProvidesPackage{notes}[1995/02/13 1.2 G. Smith]
```

`\ProvideTextCommand{\命令}{编码}[参数个数][选项]{定义} [p]` 类似于命令 `\providecommand`, 定义 `\命令`, 不过只有当编码是当前字体的编码时, 此定义才有效, 而且如果此 `\命令` 对于编码已有定义, 就不再重新定义它.

`\ProvideTextCommandDefault{\命令}{编码}[参数个数][选项]{定义}` [p] 类似于 `\DeclareTextCommandDefault`, 不过若 `\命令` 对于默认编码已有定义, 就不再重新定义它.

`\ps` 文本 用于“信件”(letter)文件类中, 在信末加入一个附言.

`\Psi` [m] 产生 Ψ .

`\psi` [m] 产生 ψ .

`\pushtabs` 在 `tabbing` 环境里, 存储当前的制表位, 它可被 `\poptabs` 命令恢复.

`\put(x,y){图形元素}` 在 `picture` 环境里把图形元素放在以 (x,y) 为参考点的位置上.

`\qbezier[点数](x1,y1)(x2,y2)(x3,y3)` 本命令应被用于 `picture` 环境内, 生成一条端点为 (x_1,y_1) , (x_3,y_3) 的二次 Bézier 曲线, 以 (x_2,y_2) 作为控制点. 当选项点数给出时, 此曲线由 (点数+1) 个点构成, 否则由程序自动确定所需的点数. 除了点数作为选项外, 其余均同 `\bezier`.

`\quad` 插入长为 1em 的水平空白.

`\qqquad` 插入长为 2em 的水平空白.

`\quotedblbase` 当 T1 编码被激活时, 产生符号 ...

`\quotesinglbase` 当 T1 编码被激活时, 产生符号 ..

`\r{x}` 产生一个重音号: `\r{o} = ô`.

`\raggedbottom` 如果没有选取选项 `twoside`, 则这是“文章”(article)、“报告”(report)与“信件”(letter)文件类的默认页格式. 在这种格式中, 段落之间的距离是固定的, 从而最后一行的位置各不相同. 与此相反的是 `\flushbottom`.

`\raggedleft` 在此命令以后的行只在右边对齐, 左边可以参差不齐, 可用 `\\` 提示换行. 参见 `\begin{flushright}`.

`\raggedright` 在此命令以后的行只在左边对齐, 右边可以参差不齐, 可用 `\\` 提示换行. 参见 `\begin{flushleft}`.

`\raisebox{升高量}[盒高][盒深]{文本}` 把包含文本的 LR 盒子从当前基线提升指定的升高量, 当升高量取负值时则是下降. 当选项盒高与盒深给出时, 则认为此盒子具有指定的高度与深度, 而不管它的实际大小如何.

`\raisetag{长度}` [m][a] 在 $\text{\textit{AMS-L}}\text{\textit{ATEX}}$ 里, 本命令把多行公式环境中的公式标号从正常位置提升指定的长度.

`\rangle` [m] 产生 \rangle .

`\rceil` [m] 产生 \rceil .

`\Re` [m] 产生 \Re .

`\ref{标记}` 打印出标记出现处的节号、公式号、图号或表号, 而标记则由命令 `\label{标记}` 设置.

`\reflectbox{文本}` 在输入宏包 `graphics` 后, 本命令使得含文本的 LR 盒子沿中心线作水平翻转(镜射).

`\refname` 在“文章”(article)文件类中含有参考文献标题的命令, 在英语中定义为‘Reference’, 但可被重新定义以适用于其他语言.

`\refstepcounter{计数器}` 类似于 `\stepcounter`, 把计数器中存储的值增加1, 但本命令也使得计数器成为 `\label-\ref` 交叉参考命令中用到的计数器.

`\renewcommand{\命令名}[参量个数][选项]{命令内容}` 类似于 `\newcommand`, 只是 `\命令名` 已经存在, 否则会显示出错信息.

`\renewcommand*{\命令名}[参量个数][选项]{命令内容}` 与 `\renewcommand` 类似, 不过命令内容必须是短的, 即在命令内容中不能出现新的段落, 也就是说, 不能有 `\par` 及空白行.

`\renewenvironment{环境名}[参量个数][选项]{进入命令}{退出命令}` 类似于 `\newenvironment`, 只是环境名已经存在, 否则会显示出错信息.

`\renewenvironment*{环境名}[参量个数][选项]{进入命令}{退出命令}` 类似于 `\renewenvironment`, 不过 `\begin{环境名}` 的参数必须是短的, 即不能出现新的段落, 也就是说, 不能有 `\par` 及空白行.

`\RequirePackage[选项]{宏包}[版本] [p]` 在类文件或宏包文件内, 本命令等价于 `\usepackage`, 它输入一个或几个以 `sty` 为扩展名的宏包文件, 如果宏包不止一个, 则可用逗号隔开, 选项被应用于所有输入的宏包文件. 此外, 命令 `\documentclass` 中列举的选项也应用于所有输入的宏包文件.

选项版本是形如 `yyyy/mm/dd` 的日期, 例如 `1994/08/01`, 如果宏包文件的日期比它早, 就会显示警告信息.

`\RequirePackageWithOptions{宏包}[版本] [p]` 类似于 `\RequirePackage`, 只是当前使用的所有选项都被应用于输入的宏包中.

`\resizebox{宽度}{高度}{文本}` 在输入宏包 `graphics` 后, 它把包含文本的 LR 盒子放缩到指定的宽度与高度, 如果这两个长度中有一个是 `!`, 则它们使用同一个放大倍数.

`\resizebox*{宽度}{高度}{文本}` 类似于 `\resizebox`, 只是高度是指包括 LR 盒子的高度与深度之和的总高度.

`\reversemarginpar` 把页边注记移到与标准位置(右侧或‘外’侧)相反的位置, 可用命令 `\normalmarginpar` 取消其作用.

`\rfloor [m]` 产生 \rfloor .

`\rhd [m]` 产生▷.

`\rho [m]` 产生 ρ .

`\right`定界符 `[m]` 调整定界符的大小使其适应夹在`\left ... \right`内的公式的高度, 例如: `\right]`. 如果另一端没有明显的定界符与之配对, 则可使用空定界符(`\left.`).

`\Rightarrow [m]` 产生 \Rightarrow .

`\rightarrow [m]` 产生 \rightarrow .

`\rightharpoondown [m]` 产生 \searrow .

`\rightharpoonup [m]` 产生 \nearrow .

`\rightleftharpoons [m]` 产生 \rightleftharpoons .

`\rightmargin` 在`list`环境里, 这是环境内文本的右边界与外部正文的右边界相比的缩进值, 正常值是`0pt`, 可用命令`\setlength`为其重置新值:

`\setlength{\rightmargin}{0.5cm}`

`\rm` 转变为罗马族, 直立形状, 中等粗细系列(Roman, upright, medium)的字体属性, 这是默认的字体.

`\rmdefault` 本命令定义由`\rmfamily`命令选取的族属性, 可用`\renewcommand`重新定义:

`\renewcommand{\rmdefault}{ptm}`

`\rmfamily` 本命令使字体保持当前的系列与形状属性, 但转变为罗马族属性.

`\Roman{计数器}` 把计数器的当前值用大写罗马数字打印出来.

`\roman{计数器}` 把计数器的当前值用小写罗马数字打印出来.

`\rotatebox{角度}{文本}` 在输入宏包`graphics`后, 本命令使得含文本的LR盒子以基线的左端点为中心, 按反时针方向旋转指定的角度.

`\rq` 产生', 同符号'.

`\rule[升高量]{宽度}{高度}` 产生一个具有指定宽度与高度的实心矩形, 当选项升高量给出时, 将此矩形从基线提升指定的升高量. 如果宽度或高度之一取`0pt`, 就能得到一条无形的线段, 它能用来产生水平或竖直空白.

`\rvert [m][a]` 产生| (右定界符).

`\rVert [m][a]` 产生|| (右定界符).

`\S` 产生§.

`\SS` 产生SS.

`\savebox{\盒子名}[宽度][位置]{文本}` 类似于`\makebox`, 只是它并不立即打印, 而是保存在`\盒子名`中, 这里的`\盒子名`必须已经用`\newsavebox`定义过. 这个盒子可利用命令`\usebox{\盒子名}`被多次使用.

`\savebox{\子图盒子}(宽度,高度)[位置]{子图图形}` 在 `picture` 环境里, 一个子图图形可被存储于具有指定宽度和高度的 `\子图盒子` 中, 这里的 `\子图盒子` 必须已经用 `\newsavebox` 定义过. 选项位置的意义同 `picture` 环境里的命令 `\makebox`, 这个盒子可利用命令 `\usebox{\盒子名}` 在 `picture` 环境里被多次使用.

`\sb` 产生一个下标, 同符号 `_`.

`\sbox{\盒子名}{文本}` 把文本保存在 LR 盒子 `\盒子名` 中, 这里的 `\盒子名` 必须已经用 `\newsavebox` 定义过. 这个盒子可利用命令 `\usebox{\盒子名}` 被多次使用.

`\sc` 转变为罗马族, 小型大写形状, 中等粗细系列 (ROMAN, CAPS AND SMALL CAPS, MEDIUM) 的字体属性.

`\scalebox{水平倍数}[竖直倍数]{文本}` 在输入宏包 `graphics` 后, 本命令使得含文本的 LR 盒子分别按水平倍数以及竖直倍数放缩, 当竖直倍数不出现时, 认为它等于水平倍数.

`\scdefault` 本命令定义由 `\scshape` 命令选取的形状属性, 可用 `\renewcommand` 重新定义:

```
\renewcommand{\scdefault}{sc}
```

`\scriptscriptstyle [m]` 在数学模式里使字体大小按 `\scriptscriptstyle` 选取.

`\scriptsize` 把字体尺寸转换成 `\scriptsize`, 它比 `\footnotesize` 小, 比 `\tiny` 大.

`\scriptstyle [m]` 在数学模式里使字体大小按 `\scriptstyle` 选取.

`\scshape` 本声明把字体的形状属性改为小型大写, 但保留族与系列不变.

`\searrow [m]` 产生 `\`.

`\sec [m]` 本命令生成数学公式里的函数名 `'sec'`.

`\section[短名]{节名}` 开始新的一节, 以当前的章号 (仅出现在“书籍”或“报告”文件类) 加上一个自动产生的节号, 再添上节名作为节的标题, 如果给出了可选项短名, 那么它将在目录以及书眉上取代原有的节名.

`\section*{节名}` 同 `\section`, 只是没有编号, 而且节名也不出现在目录中.

`\see` 配合 `Makeindex` 程序, 在文件 `makeidx.sty` 中定义的命令, 它出现在 `\index` 命令中, 表示参看索引纪录中的其它条目, 例如:

```
\index{entry|see{reference}}
```

请注意: 在上面例子中 `\` 应被 `|` 取代, 成为 `|see`.

`\seename` 在文件 `makeidx.sty` 中定义的命令, 它含有命令 `\see` 生成的文字, 在英语中定义为 `'see'`, 但可被重新定义以适用于其他语言.

`\selectfont` 激活当前的字体属性, 使它成为当前文本的字体, 它通常出现在改变字体属性的命令后面, 例如:

```
\fontshape{sl}\selectfont
```

`\selectlanguage{语言}` 出现在像 `esperant`, `german` 这样的多种语言包, 或者 `babel` 系统中, 用以改变语种. 其效果是: 标题名、命令 `\today` 所生成的日期格式、与不同语言有关的命令以及单词的拆分模式都发生变化. 例如:

```
\selectlanguage{english}
```

`\setboolean{开关}{值}` 在已输入 L^AT_EX 标准宏包 `ifthen` 的条件下, 本命令给布尔开关置值, 这里的值必须是 `true` 或 `false`, 而且此开关必须预先用命令 `\newboolean{开关}` 创建. 命令 `\boolean{开关}` 可测试此开关的值, 用于命令 `\ifthenelse` 及 `\whiledo` 的测试部分作为逻辑语句.

`\setcounter{计数器}{值}` 把整数值赋给计数器.

`\setlength{\长度命令}{长度值}` 把长度值赋给 `\长度命令`, 这里的长度值可以是固定值或弹性值.

`\SetMathAlphabet{\命令}{数学变体}{编码}{族}{系列}{形状} [p]` 对于已被 `\DeclareMathAlphabet` 命令声明过的数学模式下设置字体的命令 `\命令`, 定义特定的数学变体所应取的字体属性, 例如:

```
\DeclareMathAlphabet{\mathsl}{bold}{OT1}{cmr}{bx}{sl}
```

`\setminus [m]` 产生 `\`.

`\SetSymbolFont{字体}{数学变体}{编码}{族}{系列}{形状} [p]` 对于已被命令 `\DeclareSymbolFont` 声明过的符号字体, 定义特定的数学变体所应取的字体属性.

`\settime{秒}` 在“投影片” (`slides`) 文件类中, 如果选项 `clock` 被选中, 则以分为单位的时间标记将出现在注记的底部, 本命令把内部时钟设置为指定的秒值, 参见 `\addtime`.

`\settodepth{\长度命令}{文本}` 把文本在基线以下的深度值赋给 `\长度命令`.

`\settoheight{\长度命令}{文本}` 把文本在基线以上的高度值赋给 `\长度命令`.

`\settowidth{\长度命令}{文本}` 把文本作为 LR 盒子的宽度值赋给 `\长度命令`.

`\sf` 转变为无衬线族, 直立形状, 中等粗细系列 (`sans serif`, `upright`, `medium`) 的字体属性.

`\sfdefault` 本命令定义由 `\sffamily` 命令选取的族属性, 可用 `\renewcommand` 重新定义:

```
\renewcommand{\sfdefault}{phv}
```

`\sffamily` 本命令使字体保持当前的系列与形状属性, 但转变为无衬线族属性.

`\sharp [m]` 产生♯.

`\shortstack[位置]{文本}` 把文本竖直堆叠成一系列的格式, 其中用`\\`指示换行. 选项位置可取值`l`或`r`, 表示文本居左或居右对齐, 默认的是居中, 例如:

$$\backslash shortstack{aa\\bbb\\cc} = \begin{array}{c} aa \\ bbb \\ cc \end{array}$$

`\showhyphens{单词表}` 在终端上显示单词表中各词的可能拆分方式.

`\sideset{前置}{后置}\符号 [m][a]` 把上下标前置与后置分别放在数学\符号的前后, 例如:

$$\backslash sideset{_{\dag^*}}{_{\dag^*}}\backslash prod = \prod_{\dag}^{\dag}$$

`\Sigma [m]` 产生 Σ .

`\sigma [m]` 产生 σ .

`\signature{名字}` 用在文件类“信件”(letter)中, 打印在签名的下方, 以给出签名人的名字, 用在签名与`\name`的名字不一致的情形.

`\sim [m]` 产生 \sim .

`\simeq [m]` 产生 \simeq .

`\sin [m]` 本命令生成数学公式里的函数名‘sin’.

`\sinh [m]` 本命令生成数学公式里的函数名‘sinh’.

`\sl` 转变为罗马族, slanted的斜体形状, 中等粗细系列 (*Roman, slanted, medium*) 的字体属性.

`\sldefault` 本命令定义由`\slshape`命令选取的形状属性, 可用`\renewcommand`重新定义:

$$\backslash renewcommand{\sldefault}{sl}$$

`\sloppy` 在遇到这个命令后, 词与词之间的间隔允许大于通常的值, 使得一个段落可以分拆成更多的行, 与此相反的命令是`\fussy`, 参见`\begin{sloppypar}`,

`\slshape` 本声明把字体的形状属性改为`slanted`的斜体, 但保留族与系列不变.

`\small` 把字体尺寸转换成`\small`, 它比`\normalsize`小, 比`\footnotesize`大.

`\smallskip` 插入一个值为`\smallskipamount`的小的竖直间隔, 参见`\medskip`, `\bigskip`.

`\smallskipamount` 用于`\smallskip`的竖直间隔的标准值, 可用命令`\setlength`加以改变:

$$\backslash setlength{\smallskipamount}{1ex plus0.5ex minus0.3ex}$$

`\smile [m]` 产生 \smile .

`\sp` 产生一个上标, 同符号 \wedge .

`\spadesuit [m]` 产生♠.

`\sqcap [m]` 产生 \sqcap .

`\sqcup` [m] 产生 \sqcup .

`\sqrt[n]{变量}` [m] 本命令产生一个根号, 可选参数 n 是根指数, 如 `\sqrt[3]{2}` $= \sqrt[3]{2}$, `\sqrt{2}` $= \sqrt{2}$.

`\sqsubset` [m] 产生 \sqsubset .

`\sqsubseteq` [m] 产生 \sqsubseteq .

`\sqsupset` [m] 产生 \sqsupset .

`\sqsupseteq` [m] 产生 \sqsupseteq .

`\ss` 产生 β .

`\stackrel{上部}{下部}` [m] 把数学符号上部放在符号下部的上面, 而且使上部用较小的字体打印:

$$\stackrel{\alpha}{\longrightarrow}$$

`\star` [m] 产生 \star .

`\stepcounter{计数器}` 使计数器的数增加 1.

`\stretch{十进小数}` 是一个弹性长度, 其自然值是 0pt, 其伸展性等于十进小数乘以 `\fill`.

`\subitem{子条目}` 在 `theindex` 环境里, 本命令生成低于 `\item` 的一个二级子条目.

`\subjectname` 这是“信件”(letter)文件类的命令, 它打印出命令 `\subject` 的文字, 在英语环境里是‘Subject’, 但可被重新定义以适应其它语言.

`\subparagraph[短标题]{标题}` 章节层次中的最后一级, 位于 `\paragraph` 之下, 它在当前段落编号后面添加自动生成的小段序号, 后面再打印标题. 可选项短标题用在目录中代替标题.

`\subparagraph*{标题}` 类似于 `\subparagraph`, 只是没有编号, 也不出现在目录里.

`\subsection[短标题]{标题}` 章节层次中介于 `\section` 与 `\subsubsection` 之间, 它在当前节号的后面添加自动生成的小节序号, 后面再打印标题. 可选项短标题用在目录中代替标题.

`\subsection*{标题}` 类似于 `\subsection`, 只是没有编号, 也不在目录中出现.

`\substack{第一行\\...\\最后一行}` [m][a] 生成居中对齐的多行上下标或上下界, 它被括号 `\{...\}` 扩起来后紧跟在 \wedge 或 $-$ 的后面.

`\subsubitem{子条目}` 在 `theindex` 环境里, 本命令生成低于 `\subitem` 的一个三级子条目.

`\subsubsection[短标题]{标题}` 章节层次中介于 `\subsection` 与 `\paragraph` 之间, 它在当前小节号的后面添加自动生成的次小节序号, 后面再打印标题. 可

选项短标题用在目录中代替标题.

`\subsubsection*{标题}` 类似于 `\subsubsection`, 只是没有编号, 也不在目录中出现.

`\subset [m]` 产生 \subset .

`\subseteq [m]` 产生 \subseteq .

`\succ [m]` 产生 \succ .

`\succeq [m]` 产生 \succeq .

`\sum [m]` 产生 \sum .

`\sup [m]` 在数学公式中产生函数名 'sup' 的命令, 可带下标作为下界.

`\suppressfloats[位置]` 从这个命令至当前页结尾之间的浮动单元都被推迟到下页及以后, 选项位置可取值 t 或 b (只能取一个), 表示只推迟符合此选项的浮动单元.

`\supset [m]` 产生 \supset .

`\supseteq [m]` 产生 \supseteq .

`\surd [m]` 产生 \surd .

`\swarrow [m]` 产生 \swarrow .

`\symbol{n}` 产生当前字体中内部编号为 n 的字符.

`\t{xy}` 产生连接两个字符的重音号: `\t{oo} = \text{\textcirc{o}}`.

`\tabbingsep` 在 `tabbing` 环境里, 如果遇到文本 `\`, 命令 `\` 表示跳到当前列的末尾, 使文本右对齐, 这时 `\tabbingsep` 规定了文本与右边制表位间的留空. 可用 `\setlength` 为其重置新值.

`\tabcolsep` 在 `tabular` 环境里, 两列文本间的留空等于 `\tabcolsep` 的两倍. 可用 `\setlength` 为其重置新值:

`\setlength{\tabcolsep}{3mm}`

`\tableofcontents` 把目录打印出来.

`\tablename` 含有表格标题的命令, 在英语中定义为 'Table', 但可被重新定义以适用于其他语言.

`\tabularnewline[长度]` 在 `tabular` 或 `array` 环境里, 结束当前行, 如选项长度被给出, 则再插入指定长度的竖直间隔. 这等价于 `\\[长度]`, 不过本命令的意义更明确, 不会造成究竟是一列内部的换行还是整行的换行的含糊不清的情形. 尤其在最后一列中出现命令 `\raggedright` 时, 这时必须用 `\tabularnewline` 代替 `\\`.

`\tag{标记} [m][a]` 在 $\mathcal{A}\mathcal{M}\mathcal{S}$ -L^AT_EX 的多行公式环境里, 本命令打印出加上括号的 (标记) 以取代公式编号, 本命令带 * 号的形式不加括号.

`\tan [m]` 本命令生成数学公式里的函数名 ‘tan’.

`\tanh [m]` 本命令生成数学公式里的函数名 ‘tanh’.

`\tau [m]` 产生 τ .

`\tbinom{上部}{下部} [m][a]` 类似于 `\binom`, 产生组合数, 不过由 `\textstyle` 确定字体大小.

`\telephone{数字}` 用于 “信件” (letter) 文件类中, 输入发信人的电话号码. 在标准 L^AT_EX 的 `letter.sty` 中, 仅当 `\address` 命令不被调用时才会打印这个数字. 本命令的目的是为了公司信件格式如 `mpletter` 的需要.

`\TeX` 产生 T_EX.

`\text{短文本} [m][a]` 在输入宏包 `amsmath` 或 `amstext` 后, 本命令在数学环境里按正文格式打印短文本, 如果出现在上下标, 它会自动调节字体的大小.

`\text` 符号名 本系列命令是为了产生一些原本要用数学模式或字母组合才能得到的符号, 它们有:

`\textbullet (•); \textemdash (—); \textendash (–);`

`\textexclamdown (!); \textperiodcentered (·);`

`\textquestiondown (¿); \textquotedblleft (“);`

`\textquotedblright (”); \textquoteleft (‘);`

`\textquoteright (’); \textvisiblespace (␣);`

`\textasciicircum (^); \textasciitilde (~);`

`\textbackslash (\); \textbar (|);`

`\textgreater (>); \textless (<).`

`\textbf{文本}` 本命令等价于 `{\bfseries 文本}`, 它使用当前族与形状, 但是黑体系列 (**bold**) 的字体来打印文本.

`\textcircled{字符}` 在字符外面加一个圈: `\textcircled{s} = Ⓢ`.

`\textcolor 色彩{文本}` 宏包 `color` 中定义的命令, 它把文本用指定的色彩打印, 色彩的定义方法同 `\color`.

`\textcompwordmark` 本命令使两个字母不产生连字, 如:

`f\textcompwordmark i = fi`

`\textfloatsep` 位于顶部或底部的浮动单元与相邻的正文之间的间隔距离, 可用 `\setlength` 命令为其重置新值:

`\setlength{\textfloatsep}{20pt plus2pt minus4pt}`

`\textfraction` 在正文与浮动单元混合的页面里, 正文所占的最小比例, 其值是一个十进小数. 可用以下命令为其重置新值:

`\renewcommand{\textfraction}{0.5}`

`\textheight` 每页正文所占的高度, 不包括页眉和页底的页码, 可用 `\setlength` 命令为其重置新值:

`\setlength{\textheight}{45\baselineskip}`

`\textit{文本}` 本命令等价于 `{\itshape 文本}`, 它使用当前族与系列, 但是斜体 (*italic*) 形状的字体来打印文本.

`\textmd{文本}` 本命令等价于 `{\mdseries 文本}`, 它使用当前族与形状, 但是中等权重 (medium) 系列的字体来打印文本.

`\textnormal{文本}` 本命令等价于 `{\normalfont 文本}`, 它使用默认的族、系列与形状的字体来打印文本.

`\textquotedbl` 当 T1 编码被激活时, 产生字母 ".

`\textregistered` 产生 ®.

`\textrm{文本}` 本命令等价于 `{\rmfamily 文本}`, 它使用当前系列与形状, 但是罗马族的字体来打印文本.

`\textsc{文本}` 本命令等价于 `{\scshape 文本}`, 它使用当前族与系列, 但是小型大写 (CAPS AND SMALL CAPS) 形状的字体来打印文本.

`\textsf{文本}` 本命令等价于 `{\sffamily 文本}`, 它使用当前系列与形状, 但是无衬线 (sans serif) 族的字体来打印文本.

`\textsl{文本}` 本命令等价于 `{\slshape 文本}`, 它使用当前族与系列, 但 *slanted* 斜体形状的字体来打印文本.

`\textstyle [m]` 在数学模式里使字体大小按 `\textstyle` 选取.

`字符` 在当前文本里产生一个上标, 例如:

`ab = ab.`

`\texttrademark` 产生 TM.

`\texttt{文本}` 本命令等价于 `{\ttfamily 文本}`, 它使用当前系列与形状, 但是打字机 (typewriter) 族的字体来打印文本.

`\textup{文本}` 本命令等价于 `{\upshape 文本}`, 它使用当前族与系列, 但是直立 (upright) 形状的字体来打印文本.

`\textwidth` 页面中正文的宽度, 在双栏格式中, 这个宽度等于两个栏的宽度加上中间的留空. 可用 `\setlength` 为其重置新值.

`\tfrac{分子}{分母} [m][a]` 按 `\textstyle` 的字体大小产生一个分式.

`\TH` 当 T1 编码被激活时, 产生字母 Þ.

`\th` 当 T1 编码被激活时, 产生字母 þ.

`\thanks{脚注文本}` 当遇到命令 `\maketitle` 时, 在标题页内针对作者名产生一个内容为脚注文本的脚注.

\the计数器 把计数器的数格式化后打印的内部命令, 其输出可供其它计数器使用, 例如, **\thesubsection** 可被定义为:

\thesection.\roman{subsection}

可用 **\renewcommand{\the计数器}{定义}** 重新定义.

\Theta [m] 产生 Θ .

\theta [m] 产生 θ .

\thicklines 在 **picture** 环境里, 此命令增加斜线、箭头、圆以及圆角矩形的线条厚度, 使它们比正常情形更厚.

\thinlines 在 **picture** 环境里, 此命令重置斜线、箭头、圆以及圆角矩形的线条厚度, 使它们回复正常.

\thinspace [a] 这是 **\,** 的别名, 在数学模式里产生一个小的空白.

\thispagestyle{格式} 本命令只改变当前页的格式, 格式的取值可以是: **plain**, **empty**, **headings** 以及 **myheadings**.

\tilde{x} [m] 产生数学变量上的重音号: **\tilde{a}** = \tilde{a} .

\Tilde{x} [m][a] 本命令与 **\Tilde** 同样使用, 不过当出现 **AMS-L^AT_EX** 的多重数学重音号时, 本命令会自动调节位置.

\tiny 把字体尺寸转换成最小的 **\tiny**, 它比 **\scriptsize** 小.

\times [m] 产生 \times .

\title{文本} 遇到 **\maketitle** 命令后, 在标题页产生内容为文本的标题.

\rightarrow [m] 产生 \rightarrow .

\today 以美国方式打印当前日期, 如要改成英国式或其它语言, 则可利用 **\day**, **\month** 以及 **\year** 对其重新定义.

\top [m] 产生 \top .

\topfigrule 本命令仅当一个浮动环境出现在页顶时被执行, 它通常被定义为空, 但可被重定义以在页面的正文与浮动部分之间画一条直线, 请注意这条命令不应该产生额外的竖直空间, 例如:

\renewcommand{\topfigrule}{\vspace*{-.4pt}}

\rule{\columnwidth}{.4pt}}

\topfraction 页面中可供顶部的浮动单元使用的最大部分, 这是一个十进小数, 可用以下命令重置其值:

\renewcommand{\topfraction}{0.5}

\topmargin 从纸顶到页眉的距离, 可用 **\setlength** 为其重置新值:

\setlength{\topmargin}{0.5in}

\topnumber 可出现在一个页面顶部的浮动单元最大个数, 可用命令

`\setcounter{topnumber}{3}`

为其重置新值.

`\topsep` 在 `list` 环境的头尾, 除了 `\parskip` 外, `\topsep` 规定了额外的垂直空白.

当文件类选项 `fleqn` 被选取后, 在行间数学公式的前后也被插入这样的空白.

可用 `\setlength` 为其重置新值:

`\setlength{\topsep}{4pt plus2pt minus2pt}`

`\topskip` 从正文的顶部到正文第一行基线的距离, 可用 `\setlength` 为其重置新值:

`\setlength{\topskip}{12pt}`

`\totalheight` 一个盒子的总高度, 即高度与深度之和, 用于 `\makebox`, `\framebox` 以及 `\savebox` 的宽度参数中, 或者 `\parbox` 及 `minipage` 环境的高度参数中, 如:

`\framebox[6\totalheight]{text}`

`totalnumber` 可出现在一个页面里的浮动单元最大个数, 可用命令

`\setcounter{totalnumber}{3}`

为其重置新值.

`\triangle [m]` 产生 \triangle .

`\triangleleft [m]` 产生 \triangleleft .

`\triangleright [m]` 产生 \triangleright .

`\tt` 转变为打字机族, 直立形状, 中等粗细系列 (`typewriter`, `upright`, `medium`) 的字体属性.

`\ttdefault` 本命令定义由 `\ttfamily` 命令选取的族属性, 可用 `\renewcommand` 重新定义:

`\renewcommand{\ttdefault}{pcr}`

`\ttfamily` 使字体保持当前的系列与形状属性, 但转变为打字机 (`typewriter`) 族属性.

`\twocolumn[文本]` 另起一页并转换成双栏格式, 选项文本被排成一栏, 跨越两栏的分隔.

`\typein[命令]{信息}` 在终端屏幕上显示信息, 并等待用户输入回答, 用户回答的文本被放入选项 `命令` 中, 或在没有选项时放入 `\@typein` 内, 等用户输入回车后程序继续进行.

`\typeout{信息}` 把信息输出在终端屏幕上以及 `log` 文件中, 同时程序继续进行.

`\u{x}` 产生一个重音号: `\u{o} = ö`.

`\unboldmath` 与命令 `\boldmath` 的作用相反, 它必须在进入数学模式前出现在文

本模式中, 使得数学模式的字体恢复正常, 即数学斜体.

`\underbrace{公式} [m]` 在公式的下面产生一个水平的花括号, 以下标形式出现在本命令后面的数学符号将被居中放在花括号的下方. 例如:

$$\begin{aligned} \underbrace{a+b} &= a+b \\ \underbrace{x+y+z}_{\xi\eta\zeta} &= x+y+z \end{aligned}$$

`\underleftarrow{公式} [m][a]` 在公式的下面画一条指向左方的长箭头.

`\underleftrightharpoon{公式} [m][a]` 在公式的下面画一条左右两边都有箭头的水平线.

`\underline{文本}` 不论数学模式还是文本模式, 都在文本的下面产生一条水平直线:

$$\underline{\text{Text}} = \underline{\text{Text}}$$

`\underrightarrow{公式} [m][a]` 在公式的下面画一条指向右方的长箭头.

`\underset{字符}{\符号} [m][a]` 把字符按下标的字体放在`\符号`下方:

$$\underset{\beta}{\longrightarrow}$$

`\unitlength` 为下面的`picture`环境定义长度单位, 可用`\setlength`为其重置新值:

$$\setlength{\unitlength}{1mm}$$

`\unlhd [m]` 产生 \lhd .

`\unrhd [m]` 产生 \rhd .

`\Uparrow [m]` 产生 \Uparrow .

`\uparrow [m]` 产生 \uparrow .

`\updefault` 本命令定义了由命令`\upshape`选取的直立形状字体, 可用以下命令予以重新定义:

$$\renewcommand{\updefault}{n}$$

`\Updownarrow [m]` 产生 \Updownarrow .

`\updownarrow [m]` 产生 \updownarrow .

`\uplus [m]` 产生 \uplus .

`\upshape` 本声明把字体的形状属性改为直立, 但保留族与系列不变.

`\Upsilon [m]` 产生 Υ .

`\upsilon [m]` 产生 υ .

`\usebox{\盒子名}` 把`\盒子名`中包含的内容插入当前文本, 以此为名的盒子是由前面的命令`\newsavebox`创建的, 用命令`\sbox`或`\savebox`存储的.

`\usecounter{计数器}` 这是 `list` 环境里的命令, 它为 `\item` 命令指定一个计数器, 每次调用 `\item` 都会使这个计数器的值增加 1.

`\usefont{编码}{族}{系列}{形状}` 激活具有指定属性, 但是大小保持当前值的字体, 本命令等价于先选定字体属性, 然后执行命令 `\selectfont`.

`\usepackage[选项表]{宏包}[版本]` [p] 输入一个或几个含有附加 L^AT_EX 或 T_EX 命令的宏包文件, 文件的扩展名是 `sty`, 如有多个文件, 则用逗号隔开. 选项表被应用于所有的宏包, 此外, `\documentclass` 中指定的选项也被应用于各个宏包.

选项版本是格式为 `yyyy/mm/dd` 的日期, 例如 `1994/08/01`. 当输入的宏包文件早于此日期时会显示警告信息. 例如:

`\usepackage{bezier,ifthen}[1994/06/01]`

`\v{x}` 产生一个重音号: `\v{o} = ö`.

`\value{计数器}` 输出存储在计数器里的数值供其它命令使用, 但它并非打印一个数. 例如命令 `\setcounter{计数器1}{\value{计数器2}}` 把计数器 2 的值赋给计数器 1.

`\varepsilon` [m] 产生 ε .

`\varinjlim` [m][a] 在公式里产生 \varinjlim .

`\varliminf` [m][a] 在公式里产生 \varliminf .

`\varlimsup` [m][a] 在公式里产生 \varlimsup .

`\varphi` [m] 产生 φ .

`\varpi` [m] 产生 ϖ .

`\varprojlim` [m][a] 在公式里产生 \varprojlim .

`\varrho` [m] 产生 ϱ .

`\varsigma` [m] 产生 ς .

`\vartheta` [m] 产生 ϑ .

`\vdash` [m] 产生 \vdash .

`\vdots` [m] 产生 \vdots .

`\vec{x}` [m] 在变量 x 上加一个箭头: `\vec{a} = \vec{a}`.

`\vector(\Delta x, \Delta y){长度}` 这是 `picture` 环境里的一个图形元素, 通常作为 `\put` 或 `\multiput` 的参数, 其作用是绘制水平或竖直向量以及某些特定斜率的向量. 对于水平或竖直向量, 长度就是向量的实际长度 (以 `\unitlength` 为单位), 对于斜向量, 长度是它在 x 轴上的投影 (即水平位移). $(\Delta x, \Delta y)$ 给出了向量的斜率, 其中 Δx 与 Δy 是一对互素整数, 满足 $-4 \leq \Delta x \leq 4$, $-4 \leq \Delta y \leq 4$.

`\vee` [m] 产生 \vee .

`\verb|文本|` 在两根竖线 `|...|` 中间的内容都被不加改动地用打字机字体打印, 特殊符号以及命令都被当成普通字符输出. 除星号 `*` 以外的字符都可以取代竖线 `|` 作为开关字符, 唯一的条件是文本内不能含有此字符.

`\verb*|文本|` 类似于 `\verb`, 不过空格被打印成 `_`.

`\vfill` 竖直弹性长度, 其自然长度等于 0, 但可伸展成任意值, 它可用于以空白填满页面的一部分, 实际上这个命令是 `\vspace{\fill}` 的缩写.

`\visible` 当文件类是“投影片”(slides)时, 本命令的作用与 `\invisible` 相反, 使得文本被正常打印. 它的作用范围可持续到所在局部环境的结束, 或遇到 `\invisible` 命令, 本命令可用于生成覆盖片.

`\vline` 在 `tabular` 环境里, 本命令在表格的列条目内打印一条竖直线.

`\voffset` 输出页面与打印机设定的打印起始线之间的竖直位移, 通常打印起始线与纸顶的距离是 1in, `\voffset` 的标准值是 0pt, 可以用 `\setlength` 命令重置新值:

```
\setlength{\voffset}{-1in}
```

`\vspace{高度}` 产生具有指定高度的竖直空白, 当它位于页首或页尾时则被忽略.

`\hspace*{高度}` 产生具有指定高度的竖直空白, 即使位于页首或页尾时也不被忽略.

`\wedge [m]` 产生 \wedge .

`\whiledo{测试}{命令序列}` 输入宏包 `ifthen` 后才有效的命令, 当逻辑语句测试的值为真时, 命令序列被反复执行. 逻辑语句可以是: 关系式(用 `< = >` 连接的两个数), 奇偶性测试(`\isodd{整数}`), 两个文本的比较(`\equal{文本一}{文本二}`), 两个长度的比较(`\lengthtest{长度一 op 长度二}`, 其中 `op` 是 `< = >` 中之一)或布尔开关的测试(`\boolean{开关}`), 这里的布尔开关是由命令 `\newboolean{开关}` 创立的, 并且由命令 `\setboolean{开关}{值}` 置值, 其中的值可以为 `true` 或 `false`, 逻辑语句还可以用逻辑算子 `\and`, `\or` 以及 `\not` 联合, 并用 `\(` 与 `\)` 分组.

测试 `\isodd`, `\lengthtest` 以及 `\boolean` 是 L^AT_EX 2_ε 的新内容.

`\widehat{xyz} [m]` 在几个符号的上面打印一个加宽的 `\hat`:

$$\widehat{xyz} = \widehat{xyz}$$

`\widetilde{xyz} [m]` 在几个符号的上面打印一个加宽的 `\tilde`:

$$\widetilde{xyz} = \widetilde{xyz}$$

`\width` 等于一个盒子的自然宽度的长度参数, 它主要用于 `\makebox`, `\framebox` 或 `\savebox` 命令的宽度参数内, 或者用于 `\parbox` 及 `minipage` 环境的高度参数中, 例如:

`\framebox[2\width]{text}`

`\wp [m]` 产生 \wp .

`\wr [m]` 产生 \wr .

`\Xi [m]` 产生 Ξ .

`\xi [m]` 产生 ξ .

`\xleftarrow[下部]{上部} [m][a]` 画一个指向左方的箭头, 其上方用上标字体打印上部, 其下方用下标字体打印选项下部.

`\xrightarrow[下部]{上部} [m][a]` 画一个指向右方的箭头, 其上方用上标字体打印上部, 其下方用下标字体打印选项下部.

`\zeta [m]` 产生 ζ .

附录三 字体表 (均以 10 点字符为例)

	0	1	2	3	4	5	6	7
'00x	Γ 0	Δ 1	Θ 2	Λ 3	Ξ 4	Π 5	Σ 6	Υ 7
'01x	Φ 8	Ψ 9	Ω 10	ff 11	fi 12	fl 13	ffi 14	ffl 15
'02x	ı 16	ı 17	` 18	´ 19	˘ 20	˙ 21	˚ 22	° 23
'03x	, 24	ß 25	æ 26	œ 27	ø 28	Æ 29	Œ 30	Ø 31
'04x	- 32	! 33	" 34	# 35	\$ 36	% 37	& 38	' 39
'05x	(40) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	: 58	; 59	i 60	= 61	ı 62	? 63
'10x	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71
'11x	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
'12x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87
'13x	X 88	Y 89	Z 90	[91	" 92] 93	^ 94	˘ 95
'14x	˙ 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103
'15x	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
'16x	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119
'17x	x 120	y 121	z 122	- 123	— 124	" 125	~ 126	˘ 127

字体布局 1: cmr10. 给出了 OT1 编码方案中字符与数值的标准对应关系。

	0	1	2	3	4	5	6	7
'00x	Γ 0	Δ 1	Θ 2	Λ 3	Ξ 4	Π 5	Σ 6	Υ 7
'01x	Φ 8	Ψ 9	Ω 10	ff 11	fi 12	fl 13	ffi 14	ffl 15
'02x	ı 16	ı 17	` 18	´ 19	˘ 20	˙ 21	˚ 22	° 23
'03x	, 24	ß 25	æ 26	œ 27	ø 28	Æ 29	Œ 30	Ø 31
'04x	- 32	! 33	" 34	# 35	\$ 36	% 37	& 38	' 39
'05x	(40) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	: 58	; 59	i 60	= 61	ı 62	? 63
'10x	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71
'11x	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
'12x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87
'13x	X 88	Y 89	Z 90	[91	" 92] 93	^ 94	˘ 95
'14x	˙ 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103
'15x	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
'16x	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119
'17x	x 120	y 121	z 122	- 123	— 124	" 125	~ 126	˘ 127

字体布局 2: cmss10. OT1 编码方案. 本字体集是无衬线字体

	0	1	2	3	4	5	6	7
'00x	Γ 0	Δ 1	Θ 2	Λ 3	Ξ 4	Π 5	Σ 6	Τ 7
'01x	Φ 8	Ψ 9	Ω 10	↑ 11	↓ 12	' 13	ı 14	¿ 15
'02x	ı 16	Ј 17	` 18	´ 19	˘ 20	˙ 21	˚ 22	˛ 23
'03x	ˆ 24	ß 25	æ 26	œ 27	ø 28	Æ 29	Œ 30	Ø 31
'04x	□ 32	! 33	" 34	# 35	\$ 36	% 37	& 38	' 39
'05x	(40) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
'10x	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71
'11x	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
'12x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87
'13x	X 88	Y 89	Z 90	[91	\ 92] 93	^ 94	_ 95
'14x	‘ 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103
'15x	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
'16x	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119
'17x	x 120	y 121	z 122	{ 123	124	} 125	~ 126	¨ 127

字体布局 3: cmtt10. 与字体布局 1 的不同之处是位于 11-15, 60, 62, 92, 123, 124 和 125 号位置的符号. 本字体集是 typewriter (打字机) 字体. 对于 *italic typewriter* 字体, 还用 £ 取代 \$.

	0	1	2	3	4	5	6	7
'00x	Γ 0	Δ 1	Θ 2	Λ 3	Ξ 4	Π 5	Σ 6	Υ 7
'01x	Φ 8	Ψ 9	Ω 10	↑ 11	↓ 12	' 13	ı 14	¿ 15
'02x	ı 16	Ј 17	` 18	´ 19	˘ 20	˙ 21	˚ 22	˛ 23
'03x	ˆ 24	SS 25	Æ 26	Œ 27	Ø 28	Æ 29	Œ 30	Ø 31
'04x	ˆ 32	! 33	" 34	# 35	\$ 36	% 37	& 38	' 39
'05x	(40) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
'10x	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71
'11x	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
'12x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87
'13x	X 88	Y 89	Z 90	[91	" 92] 93	^ 94	˙ 95
'14x	‘ 96	A 97	B 98	C 99	D 100	E 101	F 102	G 103
'15x	H 104	I 105	J 106	K 107	L 108	M 109	N 110	O 111
'16x	P 112	Q 113	R 114	S 115	T 116	U 117	V 118	W 119
'17x	X 120	Y 121	Z 122	- 123	— 124	" 125	~ 126	¨ 127

字体布局 4: cmcsc10. 与字体布局 1 的不同之处是位于 11-15, 25, 60 和 62 号位置的符号. 本字体集是 SMALL CAPS 字形, 97-122 号位置是小型大写字母.

	0	1	2	3	4	5	6	7
'00x	Γ ₀	Δ ₁	Θ ₂	Λ ₃	Ξ ₄	Π ₅	Σ ₆	Υ ₇
'01x	Φ ₈	Ψ ₉	Ω ₁₀	ff ₁₁	f ₁₂	fl ₁₃	ffi ₁₄	ffl ₁₅
'02x	ı ₁₆	ſ ₁₇	` ₁₈	´ ₁₉	˘ ₂₀	˙ ₂₁	˚ ₂₂	° ₂₃
'03x	˘ ₂₄	β ₂₅	æ ₂₆	œ ₂₇	ø ₂₈	Æ ₂₉	Œ ₃₀	Ø ₃₁
'04x	˘ ₃₂	! ₃₃	" ₃₄	# ₃₅	£ ₃₆	% ₃₇	€ ₃₈	' ₃₉
'05x	(₄₀)) ₄₁	* ₄₂	+ ₄₃	, ₄₄	- ₄₅	. ₄₆	/ ₄₇
'06x	0 ₄₈	1 ₄₉	2 ₅₀	3 ₅₁	4 ₅₂	5 ₅₃	6 ₅₄	7 ₅₅
'07x	8 ₅₆	9 ₅₇	: ₅₈	; ₅₉	i ₆₀	= ₆₁	ı ₆₂	? ₆₃
'10x	@ ₆₄	A ₆₅	B ₆₆	C ₆₇	D ₆₈	E ₆₉	F ₇₀	G ₇₁
'11x	H ₇₂	I ₇₃	J ₇₄	K ₇₅	L ₇₆	M ₇₇	N ₇₈	O ₇₉
'12x	P ₈₀	Q ₈₁	R ₈₂	S ₈₃	T ₈₄	U ₈₅	V ₈₆	W ₈₇
'13x	X ₈₈	Y ₈₉	Z ₉₀	[₉₁	" ₉₂] ₉₃	^ ₉₄	· ₉₅
'14x	' ₉₆	a ₉₇	b ₉₈	c ₉₉	d ₁₀₀	e ₁₀₁	f ₁₀₂	g ₁₀₃
'15x	h ₁₀₄	i ₁₀₅	j ₁₀₆	k ₁₀₇	l ₁₀₈	m ₁₀₉	n ₁₁₀	o ₁₁₁
'16x	p ₁₁₂	q ₁₁₃	r ₁₁₄	s ₁₁₅	t ₁₁₆	u ₁₁₇	v ₁₁₈	w ₁₁₉
'17x	x ₁₂₀	y ₁₂₁	z ₁₂₂	- ₁₂₃	— ₁₂₄	" ₁₂₅	~ ₁₂₆	¨ ₁₂₇

字体布局 5: cmti10. 与字体布局 1 的不同之处是位于 36 号位置的符号(用 £ 取代了 \$). 本字体集是 *italic* 字形.

	0	1	2	3	4	5	6	7
'00x	Γ ₀	Δ ₁	Θ ₂	Λ ₃	Ξ ₄	Π ₅	Σ ₆	Υ ₇
'01x	Φ ₈	Ψ ₉	Ω ₁₀	ff ₁₁	f ₁₂	fl ₁₃	ffi ₁₄	ffl ₁₅
'02x	ı ₁₆	ſ ₁₇	` ₁₈	´ ₁₉	˘ ₂₀	˙ ₂₁	˚ ₂₂	° ₂₃
'03x	˘ ₂₄	β ₂₅	æ ₂₆	œ ₂₇	ø ₂₈	Æ ₂₉	Œ ₃₀	Ø ₃₁
'04x	˘ ₃₂	! ₃₃	" ₃₄	# ₃₅	\$ ₃₆	% ₃₇	& ₃₈	' ₃₉
'05x	(₄₀)) ₄₁	* ₄₂	+ ₄₃	, ₄₄	- ₄₅	. ₄₆	/ ₄₇
'06x	0 ₄₈	1 ₄₉	2 ₅₀	3 ₅₁	4 ₅₂	5 ₅₃	6 ₅₄	7 ₅₅
'07x	8 ₅₆	9 ₅₇	: ₅₈	; ₅₉	i ₆₀	= ₆₁	ı ₆₂	? ₆₃
'10x	@ ₆₄	A ₆₅	B ₆₆	C ₆₇	D ₆₈	E ₆₉	F ₇₀	G ₇₁
'11x	H ₇₂	I ₇₃	J ₇₄	K ₇₅	L ₇₆	M ₇₇	N ₇₈	O ₇₉
'12x	P ₈₀	Q ₈₁	R ₈₂	S ₈₃	T ₈₄	U ₈₅	V ₈₆	W ₈₇
'13x	X ₈₈	Y ₈₉	Z ₉₀	[₉₁	" ₉₂] ₉₃	^ ₉₄	· ₉₅
'14x	' ₉₆	a ₉₇	b ₉₈	c ₉₉	d ₁₀₀	e ₁₀₁	f ₁₀₂	g ₁₀₃
'15x	h ₁₀₄	i ₁₀₅	j ₁₀₆	k ₁₀₇	l ₁₀₈	m ₁₀₉	n ₁₁₀	o ₁₁₁
'16x	p ₁₁₂	q ₁₁₃	r ₁₁₄	s ₁₁₅	t ₁₁₆	u ₁₁₇	v ₁₁₈	w ₁₁₉
'17x	x ₁₂₀	y ₁₂₁	z ₁₂₂	- ₁₂₃	— ₁₂₄	" ₁₂₅	~ ₁₂₆	¨ ₁₂₇

字体布局 6: cmsl10. 本字体集是 *slanted* 字形.

	0	1	2	3	4	5	6	7
'00x	Γ 0	Δ 1	Θ 2	Λ 3	Ξ 4	Π 5	Σ 6	Υ 7
'01x	Φ 8	Ψ 9	Ω 10	α 11	β 12	γ 13	δ 14	ϵ 15
'02x	ζ 16	η 17	θ 18	ι 19	κ 20	λ 21	μ 22	ν 23
'03x	ξ 24	π 25	ρ 26	σ 27	τ 28	υ 29	ϕ 30	χ 31
'04x	ψ 32	ω 33	ε 34	ϑ 35	ϖ 36	ϱ 37	ς 38	φ 39
'05x	\leftarrow 40	\rightharpoonup 41	\rightarrow 42	\searrow 43	\prime 44	\circ 45	\triangleright 46	\triangleleft 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	\cdot 58	\circ 59	$<$ 60	$/$ 61	$>$ 62	\star 63
'10x	∂ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71
'11x	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
'12x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87
'13x	X 88	Y 89	Z 90	\flat 91	\sharp 92	\sharp 93	\smile 94	\frown 95
'14x	ℓ 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103
'15x	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
'16x	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119
'17x	x 120	y 121	z 122	\imath 123	j 124	\wp 125	\sim 126	\wedge 127

字体布局 7: **cmmi10**. OML 编码方案. 在位置 0-39 是希腊字母. 在位置 40-47, 60-64 和 123-127 是数学符号. 黑体版本 **cmmib10** 具有同样的编码.

	0	1	2	3	4	5	6	7
'00x	$-$ 0	\cdot 1	\times 2	$*$ 3	\div 4	\diamond 5	\pm 6	\mp 7
'01x	\oplus 8	\ominus 9	\otimes 10	\oslash 11	\odot 12	\bigcirc 13	\circ 14	\bullet 15
'02x	\asymp 16	\equiv 17	\subseteq 18	\supseteq 19	\leq 20	\geq 21	\preceq 22	\succeq 23
'03x	\sim 24	\approx 25	\subset 26	\supset 27	\ll 28	\gg 29	\prec 30	\succ 31
'04x	\leftarrow 32	\rightarrow 33	\uparrow 34	\downarrow 35	\leftrightarrow 36	\nearrow 37	\searrow 38	\simeq 39
'05x	\Leftarrow 40	\Rightarrow 41	\Uparrow 42	\Downarrow 43	\Leftrightarrow 44	\diagup 45	\diagdown 46	\propto 47
'06x	$/$ 48	∞ 49	\in 50	\ni 51	Δ 52	∇ 53	$/$ 54	\mid 55
'07x	\forall 56	\exists 57	\neg 58	\emptyset 59	\Re 60	\Im 61	\top 62	\perp 63
'10x	\aleph 64	\mathcal{A} 65	\mathcal{B} 66	\mathcal{C} 67	\mathcal{D} 68	\mathcal{E} 69	\mathcal{F} 70	\mathcal{G} 71
'11x	\mathcal{H} 72	\mathcal{I} 73	\mathcal{J} 74	\mathcal{K} 75	\mathcal{L} 76	\mathcal{M} 77	\mathcal{N} 78	\mathcal{O} 79
'12x	\mathcal{P} 80	\mathcal{Q} 81	\mathcal{R} 82	\mathcal{S} 83	\mathcal{T} 84	\mathcal{U} 85	\mathcal{V} 86	\mathcal{W} 87
'13x	\mathcal{X} 88	\mathcal{Y} 89	\mathcal{Z} 90	\cup 91	\cap 92	\oplus 93	\wedge 94	\vee 95
'14x	\vdash 96	\dashv 97	\lfloor 98	\rfloor 99	\lceil 100	\rceil 101	$\{$ 102	$\}$ 103
'15x	\langle 104	\rangle 105	$\ $ 106	\parallel 107	\updownarrow 108	\Updownarrow 109	\backslash 110	\wr 111
'16x	\surd 112	Π 113	∇ 114	\int 115	\sqcup 116	\sqcap 117	\sqsubseteq 118	\sqsupseteq 119
'17x	\S 120	\dagger 121	\ddagger 122	\P 123	\clubsuit 124	\diamond 125	\heartsuit 126	\spadesuit 127

字体布局 8: **cmsy10**. OMS 编码方案. 除了包含许多数学符号之外, 在位置 65-90 是 calligraphic (手写花体) 字母. 黑体版本 **cmbsy10** 具有同样的编码.

	0	1	2	3	4	5	6	7
'00x	(0)	[1]	[2]	[3]	4	5	6	7
'01x	{ 8 }	{ 9 }	< 10 >	< 11 >	12	13	/ 14 \	/ 15 \
'02x	(16)	(17)	(18)	(19)	[20]	[21]	22	23
'03x	[24]	[25]	{ 26 }	{ 27 }	< 28 >	< 29 >	/ 30 \	/ 31 \
'04x	(32)	(33)	[34]	[35]	36	37	38	39
'05x	{ 40 }	{ 41 }	< 42 >	< 43 >	/ 44 \	/ 45 \	/ 46 \	/ 47 \
'06x	(48)	(49)	[50]	[51]	52	53	54	55
'07x	(56)	(57)	(58)	(59)	{ 60 }	{ 61 }	62	63
'10x	(64)	(65)	66	67	< 68 >	< 69 >	U 70	U 71
'11x	φ 72	φ 73	⊙ 74	⊙ 75	⊕ 76	⊕ 77	⊗ 78	⊗ 79
'12x	Σ 80	Π 81	∫ 82	U 83	∩ 84	⊕ 85	Λ 86	V 87
'13x	Σ 88	Π 89	∫ 90	U 91	∩ 92	⊕ 93	Λ 94	V 95
'14x	Π 96	Π 97	~ 98	~ 99	~ 100	~ 101	~ 102	~ 103
'15x	[104]	[105]	106	107	[108]	[109]	{ 110 }	} 111
'16x	✓ 112	✓ 113	✓ 114	✓ 115	✓ 116	117	118	119
'17x	↑ 120	↓ 121	↖ 122	↗ 123	↘ 124	↙ 125	↗ 126	↘ 127

字体布局 9: cmex10. OMX 编码方案. 包含具有可变尺寸的数学符号.

	0	1	2	3	4	5	6	7
'00x	0	1	2	3	4	5	6	7
'01x	{ 8	} 9	{ 10	} 11	{ 12	} 13	{ 14	} 15
'02x	16	17	18	19	20	21	22	23
'03x	← 24	→ 25	→ 26	→ 27	28	29	30	31
'04x	← 32	→ 33	↑ 34	↓ 35	↔ 36	↗ 37	↘ 38	39
'05x	↔ 40	⇒ 41	↑↑ 42	↓↓ 43	↔ 44	↖ 45	↗ 46	47
'06x	48	∞ 49	50	51	52	53	54	55
'07x	(56) 57	(58) 59	{ 60	} 61	' 62	63
'10x	64	65	66	67	68	69	70	71
'11x	§ 72	§ 73	74	75	76	77	78	79
'12x	Σ 80	Π 81	∫ 82	83	84	85	86	87
'13x	Σ 88	Π 89	∫ 90	91	92	93	94	95
'14x	Π 96	Π 97	98	99	100	101	102	103
'15x	104	105	106	107	↕ 108	↕ 109	110	111
'16x	112	113	114	115	116	117	118	119
'17x	120	121	↙ 122	↘ 123	↘ 124	↙ 125	126	127

字体布局 10: euex10. 美国数学会设计的一种字体. 注意积分号是直立的.

	0	1	2	3	4	5	6	7
'00x	Њ 0	Љ 1	Ѝ 2	Э 3	І 4	Є 5	Ђ 6	Ћ 7
'01x	Њ 8	Љ 9	Ѝ 10	Э 11	і 12	є 13	ђ 14	ћ 15
'02x	Ю 16	Ж 17	Й 18	Ё 19	Ѹ 20	Ѳ 21	Ѕ 22	Ј 23
'03x	ю 24	ж 25	й 26	ё 27	ѵ 28	ѳ 29	ѕ 30	ј 31
'04x	“ 32	! 33	” 34	Ђ 35	“ 36	% 37	' 38	' 39
'05x	(40) 41	* 42	Ѓ 43	, 44	- 45	. 46	/ 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	: 58	; 59	« 60	1 61	» 62	? 63
'10x	“ 64	А 65	Б 66	Ц 67	Д 68	Е 69	Ф 70	Г 71
'11x	Х 72	И 73	Ј 74	К 75	Л 76	М 77	Н 78	О 79
'12x	П 80	Ч 81	Р 82	С 83	Т 84	У 85	В 86	Ш 87
'13x	Ш 88	Ы 89	З 90	[91	“ 92] 93	Ь 94	Ъ 95
'14x	“ 96	а 97	б 98	ц 99	д 100	е 101	ф 102	г 103
'15x	х 104	и 105	ј 106	к 107	л 108	м 109	н 110	о 111
'16x	п 112	ч 113	р 114	с 115	т 116	у 117	в 118	ш 119
'17x	ш 120	ы 121	з 122	— 123	— 124	№ 125	ь 126	ъ 127

字体布局 11: wncyr10. OT2 编码方案. 华盛顿大学的西里尔字体之一.

	0	1	2	3	4	5	6	7
'00x	` 0	´ 1	^ 2	~ 3	¨ 4	˘ 5	° 6	˘ 7
'01x	˘ 8	˘ 9	˘ 10	˘ 11	˘ 12	˘ 13	˘ 14	˘ 15
'02x	“ 16	” 17	„ 18	« 19	» 20	— 21	— 22	23
'03x	o 24	l 25	j 26	ff 27	fi 28	fl 29	ffi 30	ffl 31
'04x	˘ 32	! 33	" 34	# 35	\$ 36	% 37	& 38	' 39
'05x	(40) 41	* 42	+ 43	, 44	- 45	. 46	/ 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	: 58	; 59	< 60	= 61	> 62	? 63
'10x	@ 64	A 65	B 66	C 67	D 68	E 69	F 70	G 71
'11x	H 72	I 73	J 74	K 75	L 76	M 77	N 78	O 79
'12x	P 80	Q 81	R 82	S 83	T 84	U 85	V 86	W 87
'13x	X 88	Y 89	Z 90	[91	\ 92] 93	^ 94	_ 95
'14x	` 96	a 97	b 98	c 99	d 100	e 101	f 102	g 103
'15x	h 104	i 105	j 106	k 107	l 108	m 109	n 110	o 111
'16x	p 112	q 113	r 114	s 115	t 116	u 117	v 118	w 119
'17x	x 120	y 121	z 122	{ 123	124	} 125	~ 126	- 127
'20x	Ä 128	Å 129	Č 130	Ć 131	Đ 132	È 133	É 134	Ğ 135
'21x	Ĺ 136	Ł 137	Ł 138	Ń 139	Ň 140	Đ 141	Ō 142	Ř 143
'22x	Ŕ 144	Š 145	Š 146	Ş 147	Ť 148	Ţ 149	Ů 150	Ű 151
'23x	Ÿ 152	Ž 153	Ž 154	Ž 155	IJ 156	İ 157	đ 158	§ 159
'24x	ă 160	ą 161	ć 162	č 163	ď 164	ě 165	ę 166	ğ 167
'25x	í 168	ï 169	ł 170	ń 171	ň 172	ŋ 173	õ 174	ř 175
'26x	ř 176	ś 177	š 178	ş 179	ť 180	ţ 181	ů 182	ű 183
'27x	ÿ 184	ž 185	ž 186	ž 187	ij 188	ı 189	ı 190	£ 191
'30x	À 192	Á 193	Â 194	Ã 195	Ä 196	Å 197	Æ 198	Ç 199
'31x	È 200	É 201	Ê 202	Ë 203	Ì 204	Í 205	Î 206	Ï 207
'32x	Ð 208	Ñ 209	Ò 210	Ó 211	Ô 212	Õ 213	Ö 214	Œ 215
'33x	Ø 216	Ù 217	Ú 218	Û 219	Ü 220	Ý 221	Þ 222	ŠS 223
'34x	à 224	á 225	â 226	ã 227	ä 228	å 229	æ 230	ç 231
'35x	è 232	é 233	ê 234	ë 235	ì 236	í 237	î 238	ï 239
'36x	ð 240	ñ 241	ò 242	ó 243	ô 244	õ 245	ö 246	œ 247
'37x	ø 248	ù 249	ú 250	û 251	ü 252	ý 253	þ 254	ß 255

字体布局 12: ecrm1000. T1 编码方案.

	0	1	2	3	4	5	6	7
'00x	˘ 0	˙ 1	ˆ 2	˜ 3	¨ 4	˘ 5	˚ 6	ˇ 7
'01x	˘ 8	˙ 9	˙ 10	˙ 11	˙ 12	˙ 13	˙ 14	˙ 15
'02x	˙ 16	˙ 17	˙ 18	˙ 19	˙ 20	˙ 21	˙ 22	˙ 23
'03x	← 24	→ 25	ˆ 26	ˆ 27	ˆ 28	ˆ 29	ˆ 30	ˆ 31
'04x	h 32	33	34	35	\$ 36	37	38	' 39
'05x	40	41	* 42	43	˙ 44	= 45	˙ 46	/ 47
'06x	0 48	1 49	2 50	3 51	4 52	5 53	6 54	7 55
'07x	8 56	9 57	58	59	< 60	— 61	> 62	63
'10x	64	65	66	67	68	69	70	71
'11x	72	73	74	75	76	77	78	○ 79
'12x	80	81	82	83	84	85	86	Ω 87
'13x	88	89	90	∥ 91	92	∥ 93	↑ 94	↓ 95
'14x	˘ 96	97	★ 98	o/o 99	† 100	101	102	103
'15x	104	105	106	107	108	∞ 109	♪ 110	111
'16x	112	113	114	115	116	117	118	119
'17x	120	121	122	123	124	125	˘ 126	= 127
'20x	˘ 128	˘ 129	˘ 130	˘ 131	† 132	† 133	∥ 134	‰ 135
'21x	• 136	°C 137	\$ 138	e 139	f 140	C 141	W 142	N 143
'22x	G 144	P 145	£ 146	R 147	† 148	† 149	d 150	™ 151
'23x	‰ 152	¶ 153	B 154	Nº 155	% 156	C 157	o 158	SM 159
'24x	{ 160	} 161	¢ 162	£ 163	⊠ 164	Y 165	166	§ 167
'25x	˘ 168	© 169	ª 170	© 171	¬ 172	® 173	® 174	˘ 175
'26x	˚ 176	± 177	² 178	³ 179	˘ 180	ll 181	¶ 182	˙ 183
'27x	※ 184	¹ 185	º 186	√ 187	¼ 188	½ 189	¾ 190	€ 191
'30x	192	193	194	195	196	197	198	199
'31x	200	201	202	203	204	205	206	207
'32x	208	209	210	211	212	213	× 214	215
'33x	216	217	218	219	220	221	222	223
'34x	224	225	226	227	228	229	230	231
'35x	232	233	234	235	236	237	238	239
'36x	240	241	242	243	244	245	÷ 246	247
'37x	248	249	250	251	252	253	254	255

字体布局 13: tcrm1000. TS1 编码方案.

参考文献与网站

Knuth D. E. *The T_EXbook*, Addison-Wesley, 1984

Knuth D. E. *The METAFONT book*, Addison-Wesley, 1986

Kopka H., Daly P. W. *A Guide to L^AT_EX*, Third edition, Addison-Wesley, 1999

Lamport L. *L^AT_EX—A Document Preparation System*, Addison-Wesley, 1985

Salomon D. *The Advanced T_EXbook*, Springer-Verlag, 1995

Seroul R., Levy S. *A Beginner's Book of T_EX*, Springer-Verlag, 1991

Spivak M. D. *The Joy of T_EX*, Second edition, American Mathematical Society, 1990

von Bechtolsheim S. *T_EXin Practice*, vol. I – IV, Springer-Verlag, 1993

国内有关 T_EX 的网站有:

<http://162.105.195.105/texhome/>

<http://ctex.dhs.org>

内容相当丰富, 值得一看, 还有 C_T_EX 与 CCT 包, 可供下载.

国外的网站有:

<http://www.tug.org>

这是 T_EX 用户协会 (T_EX Users Group) 的主页, 有各种最新消息以及 T_EX 有关材料的下载.

要下载 T_EX 的各种最新版本, 可到 CTAN (Comprehensive T_EXArchive Network) 用匿名 ftp 下载, CTAN 站点的地址是:

<ftp://cam.catn.org/tex-archive/> (英国)

<ftp://dante.ctan.org/tex-archive/> (德国)

<ftp://tug.ctan.org/tex-archive/> (美国)

此外, 以下站点的‘L^AT_EX 百科全书’也是值得一看的地方:

<http://tex.loria.fr/english/index.html>

索引

- \!, 38, 172, 258
- !, 258
- ! ', 14, 258
- \", 15, 258
- \", 13, 258
- \#, 11, 258
- #, 258
- ##, 258
- \\$, 11, 259
- \$, 39, 258
- \$\$, 40
- \%, 11, 259
- %, 259
- \&, 259
- &, 259
- \', 15, 259
- \', 13
- \(, 39, 259
- (), 259
- \), 39, 259
- *, 12
- \+, 134, 259
- \,, 13, 18, 38, 172, 259
- \-, 17, 134, 259
- , 13, 259
- , 13
- , 13
- \., 15, 259
- \/, 14, 259
- \:, 38, 172, 259
- \;, 38, 172, 259
- \<, 11, 259
- \=, 15, 30, 259
- \>, 11, 30, 260
- ? ', 14, 260
- \@, 14, 260
- @, 210, 260
- \[, 40, 55, 260
- [], 260
- \\, 11, 16, 33, 260
- *, 16, 260
- \], 40, 55, 260
- \^, 11, 15, 260
- ^, 41, 260
- _, 11, 260
- _, 41, 260
- \', 15, 135, 260
- \', 13
- \{, 11, 260
- { }, 260
- \', 135
- \|, 11, 51, 64, 260
- |, 33, 260
- \}, 11, 261

-
- \~, 11, 14, 15, 261
 - ~, 212, 261
 - 10pt, 67, 281
 - 11pt, 67, 281
 - 12pt, 67, 281
 - _, 12, 14, 38, 258
 - \a', 261
 - a4paper, 67, 281
 - a5paper, 67, 281
 - \a=, 261
 - \a', 261
 - \AA, 14, 261
 - \aa, 14, 261
 - \abovedisplayshortskip, 148, 261
 - \abovedisplayskip, 148, 261
 - abstract, 72
 - \abstractname, 73, 261
 - \Acrobatmenu, 227
 - acurve (T_ydraw), 100
 - \Acute, 166
 - \acute, 48, 261
 - \addButton, 227
 - \addcontentsline, 205, 261
 - \address, 261
 - \addtime, 231, 261
 - \addtocontents, 205, 261
 - \addtocounter, 77, 261
 - \addtolength, 261
 - \addvspace, 261
 - \AE, 14, 261
 - \ae, 14, 262
 - \aleph, 64, 262
 - align, 153, 159, 264
 - align*, 153, 160
 - alignat, 153, 160, 264
 - alignat*, 153, 160
 - aligned, 161
 - aligned, 161, 264
 - \allowdisplaybreaks, 262
 - alltt, 264
 - \Alph, 124, 262
 - \alph, 124
 - Alpha, 203
 - \alpha, 60, 262
 - alpha, 203
 - \alsoname, 262
 - \amalg, 62, 262
 - amsbsy, 153
 - amscd, 175
 - amsmath, 153
 - amsopen, 153
 - amstext, 153
 - \and, 71, 262
 - \angle, 64, 262
 - angle, 222
 - appendix, 264
 - \appendixname, 262
 - \approx, 62, 262
 - \arabic, 124, 262
 - arabic, 203
 - \arccos, 65, 262
 - \arcsin, 262
 - \arctan, 65, 262
 - \arg, 65, 262
 - array, 53, 137, 143, 264
 - \arraycolsep, 58, 137, 148, 262
 - \arrayrulewidth, 137, 262

-
- `\arraysolsep`, 78
 - `\arraystretch`, 137, 262
 - `arrowhead` (`Tydraw`), 99
 - `arrowheadd` (`Tydraw`), 99
 - `article`, 10, 66
 - `\ast`, 62, 262
 - `\asymp`, 62, 262
 - `\AtBeginDocument`, 262
 - `\AtEndDocument`, 263
 - `\AtEndOfClass`, 263
 - `\AtEndOfPackage`, 263
 - `\atop`, 50, 154
 - `\author`, 70, 263
 -
 - `\b`, 15, 263
 - `b`, 33
 - `b5paper`, 67, 281
 - `\backmatter`, 203, 263
 - `\backslash`, 51, 64, 263
 - `\Bar`, 166, 263
 - `\bar`, 48, 263
 - `\baselineskip`, 22, 27, 78, 263
 - `\baselinestretch`, 22, 27, 263
 - `bb`, 223
 - `\begin{abstract}`, 263
 - `\begin{align}`, 264
 - `\begin{alignat}`, 264
 - `\begin{aligned}`, 264
 - `\begin{alltt}`, 264
 - `\begin{appendix}`, 264
 - `\begin{array}`, 264
 - `\begin{Bmatrix}`, 264
 - `\begin{bmatrix}`, 264
 - `\begin{cases}`, 264
 - `\begin{center}`, 264
 - `\begin{description}`, 264
 - `\begin{displaymath}`, 264
 - `\begin{document}`, 265
 - `\begin{enumerate}`, 265
 - `\begin{eqnarray}`, 265
 - `\begin{eqnarray*}`, 265
 - `\begin{equation}`, 265
 - `\begin{falign}`, 265
 - `\begin{figure}`, 265
 - `\begin{figure*}`, 265
 - `\begin{filecontents}`, 265
 - `\begin{filecontents*}`, 265
 - `\begin{flushleft}`, 265
 - `\begin{flushright}`, 265
 - `\begin{gather}`, 265
 - `\begin{gathered}`, 266
 - `\begin{itemize}`, 266
 - `\begin{letter}`, 266
 - `\begin{list}`, 266
 - `\begin{lrbox}`, 266
 - `\begin{math}`, 266
 - `\begin{matrix}`, 266
 - `\begin{minipage}`, 266
 - `\begin{multicols}`, 266
 - `\begin{multline}`, 267
 - `\begin{note}`, 267
 - `\begin{overlay}`, 267
 - `\begin{picture}`, 267
 - `\begin{quotation}`, 267
 - `\begin{quote}`, 267
 - `\begin{slide}`, 267
 - `\begin{sloppypar}`, 267
 - `\begin{split}`, 267
 - `\begin{subarray}`, 267

- \begin{subequations}, 267
- \begin{tabbing}, 267
- \begin{table}, 268
- \begin{table*}, 268
- \begin{tabular}, 268
- \begin{tabular*}, 268
- \begin{thebibliography}, 268
- \begin{theindex}, 268
- \begin{titlepage}, 268
- \begin{trivlist}, 268
- \begin{verbatim}, 268
- \begin{verbatim*}, 269
- \begin{verse}, 269
- \begin{Vmatrix}, 269
- \begin{vmatrix}, 269
- \begin{定理环境名}, 268
- \begin{环境名}, 263
- \begin{命令名}, 264
- \belowdisplayshortskip, 148, 269
- \belowdisplayskip, 148, 269
- \beta, 60, 269
- \bezier, 93, 269
- \bf, 269
- \bfdefault, 185, 269
- \bfseries, 25, 183, 269
- \biaosong, 216
- \bibitem, 29, 269
- \bibliography, 269
- \bibliographystyle, 269
- \bibname, 269
- \Big, 51, 270
- \big, 51, 270
- \bigcap, 64, 270
- \bigcirc, 62, 270
- \bigcup, 270
- \Bigg, 51, 270
- \bigg, 51, 270
- \Biggl, 53, 270
- \biggl, 53, 270
- \Biggm, 53, 270
- \biggm, 53, 270
- \Biggr, 53, 270
- \biggr, 53, 270
- \Bigl, 53, 270
- \bigl, 53, 270
- \Bigm, 53, 270
- \bigm, 53, 270
- \bigodot, 64, 270
- \bigoplus, 64, 270
- \bigotimes, 64, 270
- \Bigr, 53, 270
- \bigr, 53, 270
- \bigskip, 21, 270
- \bigskipamount, 21, 78, 270
- \bigsqcup, 64, 270
- \bigsup, 64
- \bigtriangledown, 62, 270
- \bigtriangleup, 62, 270
- \biguplus, 64, 271
- \bigvee, 64, 271
- \bigwedge, 64, 271
- \binom, 154, 169, 271
- Blinds, 229
- bm, 37, 61, 154
- Bmatrix, 153, 163, 264
- bmatrix, 163, 264
- \bmod, 65, 271
- bmp, 104, 105

-
- \boldmath, 60, 271
 - \boldsymbol, 61, 154, 271
 - book, 10, 66
 - \bot, 64, 271
 - \botfigrule, 271
 - \bottombuttons, 228
 - \bottomfraction, 271
 - bottomnumber, 271
 - \bowtie, 62, 271
 - \Box, 62, 64, 271
 - Box, 229
 - box (T_ydraw), 99
 - \Boxed, 271
 - \boxed, 157
 - bp, 7
 - \Breve, 166, 271
 - \breve, 48, 271
 - \bullet, 62, 271
 - \c, 15
 - c, 33
 - \c{x}, 271
 - \cal, 271
 - \cap, 62, 271
 - \caption, 271
 - \captions, 272
 - cases, 162, 264
 - \cc, 272
 - cc, 7
 - \ccdp, 216
 - \ccht, 216
 - \ccname, 272
 - \ccnospace, 217
 - \ccwd, 216
 - \cdot, 62, 272
 - \cdots, 38, 272
 - center, 28, 264
 - \centerbmp (PCTeX32), 106
 - \centereps (PCTeX32), 106
 - \centering, 28, 272
 - \centerline, 28, 272
 - \centerps (PCTeX32), 106
 - centertags, 153
 - \centerwmf (PCTeX32), 106
 - \cfrac, 168, 272
 - \changeoverlay, 227
 - \chapter, 68, 272
 - chapter, 69, 77
 - \chapter*, 272
 - \chaptername, 272
 - \Check, 166, 272
 - \check, 48, 272
 - \CheckCommand, 272
 - \CheckCommand*, 272
 - \chi, 60, 272
 - \ChineseScale (天元), 10
 - \choose, 50, 154
 - \circ, 62, 272
 - \circle, 88, 272
 - circle (T_ydraw), 99
 - \circle*, 88, 273
 - \cite, 29, 273
 - CJK, 211
 - CJK*, 211
 - \CJKboldshift, 214
 - \CJKenc, 212
 - \CJKfamily, 212
 - \CJKglue, 214
 - \CJKkern, 214

- \CJKnospace, 212
- \CJKspace, 212
- \CJKtilde, 212
- \ClassError, 273
- \ClassInfo, 273
- \ClassWarning, 273
- \ClassWarningNoLine, 273
- \cleardoublepage, 273
- \clearpage, 273
- \cline, 34, 273
 - clip, 223
- \closing, 273
- \clubsuit, 64, 273
 - cm, 7
- \colon, 38
- \color, 221, 273
 - color, 107, 220
- \colorbox, 221, 273
- \columnsep, 197, 273
- \columnseprule, 197, 274
- \columnwidth, 197
- \cong, 62, 274
- \contentsline, 274
- \contentsname, 274
- \convertMPtoPDF, 224
- \coprod, 64, 274
- \copyright, 12, 274
- \cos, 65, 274
- \cosh, 65, 274
- \cot, 65, 274
- \coth, 65, 274
- \CS, 217
- \csc, 65, 274
- \cup, 62, 274
- \CurrentOption, 274
 - curve (Tydraw), 100
- \d, 15, 274
- \dag, 12, 274
- \dagger, 62, 274
- \dashbox, 90, 274
- \dashv, 62, 64, 274
- \date, 70, 274
- \dbinom, 169, 275
- \dblfigrule, 275
- \dblfloatpagefraction, 275
- \dblfloatsep, 275
- \dbltextfloatsep, 275
- \dbltopfraction, 275
 - dbltopnumber, 275
- dd, 7
- \ddag, 12, 275
- \ddagger, 62, 275
- \ddddot, 166
- \dddot, 166
- \Ddot, 166
- \ddot, 48, 275
- \ddots, 38
- \DeclareErrorFont, 190, 275
- \DeclareFixedFont, 185, 275
- \DeclareFontEncoding, 189, 275
- \DeclareFontEncodingDefault, 189
- \DeclareFontEncodingDefaults,
 - 276
- \DeclareFontFamily, 190, 276
- \DeclareFontShape, 190, 276
- \DeclareFontSubstitution, 190,
 - 276
- \DeclareGraphicsExtensions, 276

-
- \DeclareGraphicsRule, 276
 - \DeclareMathAccent, 188, 276
 - \DeclareMathAlphabet, 186, 276
 - \DeclareMathDelimiter, 188, 277
 - \DeclareMathOperator, 171, 277
 - \DeclareMathOperator*, 171
 - \DeclareMathRadical, 188, 277
 - \DeclareMathSizes, 189, 277
 - \DeclareMathSymbol, 188, 277
 - \DeclareMathVersion, 277
 - \DeclareOldFontCommand, 186, 278
 - \DeclareOption, 278
 - \DeclareOption*, 278
 - \DeclareRobustCommand, 278
 - \DeclareRobustCommand*, 278
 - \DeclareSymbolFont, 187, 278
 - \DeclareSymbolFontAlphabet, 187, 278
 - \DeclareTextAccent, 193, 278
 - \DeclareTextAccentDefault, 193, 278
 - \DeclareTextCommand, 192, 278
 - \DeclareTextCommandDefault, 193, 279
 - \DeclareTextComposite, 193, 279
 - \DeclareTextCompositeCommand, 193, 279
 - \DeclareTextFontCommand, 185, 279
 - \DeclareTextSymbol, 193, 279
 - \DeclareTextSymbolDefault, 193, 279
 - \definecolor, 220, 279
 - \deg, 65, 279
 - \DeleteShortVerb, 279
 - \Delta, 60, 279
 - \delta, 60, 280
 - \depth, 129, 280
 - description, 122, 264
 - \det, 65, 280
 - \dfrac, 168, 280
 - \DH, 280
 - \dh, 280
 - diagrams, 177
 - \Diamond, 62, 64, 280
 - \diamond, 62, 280
 - \diamondsuit, 64, 280
 - \dim, 65, 280
 - \discretionary, 280
 - \displaybreak, 280
 - displaymath, 40, 55, 265
 - \displaystyle, 37, 149, 280
 - Dissolve, 229
 - \div, 62, 280
 - \DJ, 280
 - \dj, 280
 - document, 265
 - \documentclass, 10, 66, 280
 - \documentstyle, 281
 - \Dot, 166, 281
 - \dot, 48, 281
 - \doteq, 62, 281
 - \dotfill, 54, 281
 - \dots, 167, 281
 - \dotsb, 167, 281
 - \dotsc, 167, 281
 - \dotsi, 167, 281
 - \dotsm, 167, 281
 - dotted (Tydraw), 100

-
- \doublerulesep, 137
 - \Downarrow, 51, 63, 281
 - \downarrow, 51, 63, 281
 - draft, 68, 107, 223, 281
 - \draw (Tydraw), 99
 - dvipdf, 107
 - dvips, 107
 - dvipsone, 107
 - dviwin, 107
 - dviwindo, 107
 - \ell, 64, 281
 - \em, 25, 281
 - em, 7
 - emf, 105
 - \emph, 25, 281
 - empty, 200
 - \emptyset, 64, 281
 - emtex, 107
 - \encl, 281
 - \enclname, 281
 - \encodingdefault, 185
 - \end, 282
 - \EndHPage, 235
 - \EndLink, 236
 - \EndPreamble, 235
 - \enlargethispage, 18, 282
 - \enlargethispage*, 18, 282
 - \ensuremath, 282
 - enumerate, 122, 265
 - enumi, 123
 - enumii, 123
 - enumiii, 123
 - enumiv, 123
 - eps, 105, 108
 - \epsilon, 60, 282
 - epstopdf, 222
 - \eqnarray, 56
 - eqnarray, 40, 153, 265
 - \eqnarray*, 57
 - eqnarray*, 40, 265
 - \eqno, 40, 145
 - \eqref, 161, 282
 - equation, 40, 55, 56, 77, 153, 265
 - equation*, 153
 - \equiv, 62, 282
 - \eta, 60, 282
 - \evensidemargin, 282
 - ex, 7
 - \ExecuteOptions, 282
 - executivepaper, 67, 281
 - \exists, 64, 282
 - \ExitHPage, 235
 - \exp, 65, 282
 - \extracolsep, 282
 - falign, 265
 - \familydefault, 185
 - \fangsong, 216
 - \fbox, 130, 282
 - \fboxrule, 130, 282
 - \fboxsep, 130, 282
 - \fcolorbox, 221, 282
 - figure, 114, 265
 - figure*, 114, 265
 - \figurename, 283
 - filecontents, 265
 - filecontents*, 265
 - \fill, 8, 19, 283

- fill (Tydraw), 99
final, 68, 108, 281
flalign, 153, 160
flalign*, 153, 160
\flat, 64, 283
fleqn, 199, 281
fleqno, 154
\floatpagefraction, 283
\floatsep, 283
\flushbottom, 199, 283
flushleft, 119, 265
flushright, 119, 265
\fnsymbol, 140, 283
\fontencoding, 180
\fontfamily, 180, 283
\fontseries, 180, 283
\fontshape, 180, 283
\fontsize, 180, 283
\footnotetext, 141
\footnote, 36, 141, 283
footnote, 77, 140
\footnotemark, 141, 283
\footnoterule, 283
\footnotesep, 284
\footnotesize, 26, 36, 284
\footnotetext, 284
\footskip, 284
\forall, 64, 284
\foreignlanguage, 284
\frac, 43, 284
\frame, 284
\framebox, 90, 130, 284
frames, 235
\frenchspacing, 284
\frontmatter, 203, 284
\frown, 62, 284
\fussy, 284

\Gamma, 60, 285
\gamma, 60, 285
gather, 153, 158, 266
gather*, 153
gathered, 161, 266
\gcd, 65, 285
\ge, 62, 285
\genfrac, 169, 285
\geq, 62, 285
\gets, 63, 285
\gg, 62, 285
Glitter, 229
\glossary, 285
\glossaryentry, 285
graphics, 107
graphicx, 107, 222
\graphpaper, 285
\Grave, 166, 285
\grave, 48, 285
\guillemotleft, 285
\guillemotright, 285
\guilsinglleft, 285
\guilsinglright, 285

\H, 15, 285
\Hat, 166, 285
\hat, 48, 285
\hbar, 64, 285
\HCode, 235
\hdotsfor, 167
\headheight, 285

- \headings, 68
- headings, 200, 231
- \headsep, 285
- \headtoname, 286
- \heartsuit, 64, 286
- \height, 129, 286
- height, 222
- \heiti, 216
- \hfil, 29
- \hfill, 19, 286
- hide (Tydraw), 100
- hiderotate, 108
- hidescale, 108
- hiresbb, 108, 223
- \hline, 33, 286
- \hoffset, 286
- \hom, 65, 286
- \hookleftarrow, 63, 286
- \hookrightarrow, 63, 286
- \HPage, 235
- \href, 228
- \hrulefill, 286
- \hspace, 18, 38, 286
- \hspace*, 18, 286, 317
- html, 234
- \Huge, 26, 286
- \huge, 26, 286
- \hyperlink, 228
- \hyphenation, 286
- \i, 15, 286
- \idotsint, 165, 286
- \iff, 63, 286
- \IfFileExists, 286
- \ifHtml, 236
- \iflanguage, 286
- \ifthenelse, 286
- \iiiint, 165, 288
- \iiint, 165, 288
- \iint, 165, 288
- \Im, 64, 287
- \imageButton, 227
- ImageMagick, 108
- \imath, 64, 287
- \in, 62, 287
- in, 7
- \include, 206, 287
- \includegraphics, 109, 222, 287
- \includegraphics*, 109, 287
- \includeonly, 206, 287
- \indent, 287
- indentfirst, 21
- \index, 210, 287
- \indexentry, 288
- \indexname, 210, 288
- \indexspace, 288
- \inf, 65, 288
- \infty, 64, 288
- \input, 205, 288
- \InputIfFileExists, 288
- \insuremath, 76
- \int, 64, 288
- \intertext, 155, 288
- \intertextsep, 288
- intllimits, 154
- \invisible, 230, 288
- \iota, 60, 288
- \it, 288
- \itdefault, 185, 288

- `\item`, 123, 289
`\itemindent`, 127, 289
 `itemize`, 122, 266
`\itemsep`, 127, 289
`\itshape`, 25, 183, 289

`\j`, 15, 289
`\jmath`, 64, 289
`\Join`, 64, 289
`\jot`, 148, 289

`\k`, 289
`\kaishu`, 216
`\kappa`, 60, 289
 `keepaspectratio`, 222
`\ker`, 65, 289
`\kill`, 31, 289

`\L`, 14, 289
`\l`, 14, 289
 `l`, 33
`\label`, 56, 208, 289
`\labelenumi`, 123
`\labelenumii`, 123
`\labelenumiii`, 123
`\labelenumiv`, 123
`\labelitemn`, 289
`\labelitemi`, 123
`\labelitemii`, 123
`\labelitemiii`, 123
`\labelitemiv`, 123
`\labelnumn`, 289
`\labelsep`, 127, 290
`\labelwidth`, 127, 290
`\Lambda`, 60, 290
`\lambda`, 60, 290
 `landscape`, 67, 281
`\langle`, 51, 290
`\language`, 290
`\LARGE`, 26, 290
`\Large`, 26, 290
`\large`, 26, 290
 \LaTeX , 4
`\LaTeX`, 12, 290
`\LaTeXe`, 12, 290
 `latexsym`, 37, 61
 `lattice (Tydraw)`, 100
`\lceil`, 51, 290
`\ldots`, 38, 290
`\le`, 62, 290
`\leadsto`, 63, 290
`\left`, 52, 290
`\left.`, 52
`\Leftarrow`, 63, 290
`\leftarrow`, 63, 290
`\lefteqn`, 147, 290
`\leftharpoondown`, 63, 290
`\leftharpoonup`, 63, 290
`\leftmargin`, 127, 290
`\Leftrightarrow`, 63, 291
`\leftrightarrow`, 63, 291
`\leftroot`, 172, 291
 `legalpaper`, 67, 281
`\leq`, 62, 291
`\leqno`, 40, 145
 `leqno`, 154, 199, 281
 `letter`, 66, 266
 `letterpaper`, 67, 281
`\lfloor`, 51, 291
`\lg`, 65, 291

-
- \lhd, 62, 291
 - \lim, 65, 291
 - \liminf, 65, 291
 - \limits, 46, 154, 291
 - \limsup, 65, 291
 - \line, 86, 291
 - line (Tydraw), 99
 - \linebreak, 17, 291
 - lines (Tydraw), 99
 - \linethickness, 87, 291
 - \Link, 236
 - list, 125, 266
 - \listfigurename, 291
 - \listfiles, 205, 291
 - \listoffigures, 204, 291
 - \listoftables, 204, 291
 - \listparindent, 127, 291
 - \listtablename, 292
 - \ll, 62, 292
 - \ln, 65, 292
 - \LoadClass, 292
 - \location, 292
 - \log, 65, 292
 - \Longleftarrow, 63, 292
 - \longleftarrow, 63, 292
 - \Longleftrightarrow, 63, 292
 - \longleftrightarrow, 63, 292
 - \longmapsto, 63, 292
 - \Longrightarrow, 63, 292
 - \longrightarrow, 63, 292
 - \lq, 292
 - lrbox, 131, 266
 - \lVert, 292
 - \lvert, 292
 - \maginparpush, 293
 - \maginparsep, 293
 - \maginparwidth, 293
 - \mainmatter, 203, 292
 - \makebox, 90, 92, 130, 292
 - \makeglossary, 293
 - makeidx, 209
 - \makeindex, 209, 293
 - \makelabel, 293
 - \makelabels, 293
 - \MakeShortVerb, 293
 - \maketitle, 70, 293
 - \mapsto, 63, 293
 - \marginpar, 142, 293
 - \marginparpush, 142
 - \marginparsep, 142
 - \marginparwidth, 142
 - \margins, 227
 - \markboth, 202, 293
 - \markright, 202, 293
 - math, 39, 266
 - \mathalpha, 188
 - \mathbf, 59, 186, 293
 - \mathbin, 188
 - \mathcal, 59, 186, 293
 - \mathclose, 188
 - \mathindent, 148, 154, 199, 293
 - \mathit, 38, 59, 186, 294
 - \mathnormal, 60, 294
 - \mathop, 188
 - \mathopen, 188
 - \mathord, 188
 - \mathpunct, 188
 - \mathrel, 188

- `\mathring`, 48, 294
`\mathrm`, 47, 59, 186, 294
`\mathsf`, 59, 186, 294
`\mathstrut`, 45
`\mathtt`, 59, 186, 294
`\mathversion`, 294
 `matrix`, 163, 266
`\max`, 65, 294
 `MaxMatrixCols`, 163
`\mbox`, 17, 39, 130, 294
`\mddefault`, 185, 294
`\mdseries`, 25, 183, 294
`\medskip`, 21, 294
`\medskipamount`, 21, 78, 294
`\medspace`, 172, 294
`\MessageBreak`, 294
 `MetaPost`, 116
`\mho`, 64, 294
`\mid`, 62, 294
`\min`, 65, 294
 `minipage`, 133, 266
`\mit`, 294
 `mm`, 7
`\mod`, 294
`\models`, 62, 295
 `monochrome`, 108
`\mp`, 62, 295
`\mspace`, 172, 295
`\mu`, 60, 295
 `multicol`, 198
 `multicols`, 266
`\multicolumn`, 33, 295
`\multipt`, 86, 295
 `multline`, 153, 156, 267
 `multline*`, 153
`\multlinegap`, 156, 295
 `myheadings`, 200
`\nabla`, 64, 295
`\name`, 295
 `namelimits`, 154
`\natural`, 64, 295
`\nbs`, 213
`\ne`, 62
`\nearrow`, 63, 295
`\NeedsTeXFormat`, 193
`\NeedsTextFormat`, 295
`\neg`, 64, 295
`\negmedspace`, 172, 295
`\negthickspace`, 172, 295
`\negthinspace`, 172, 295
`\neq`, 62, 295
 `net (Tydraw)`, 100
`\newboolean`, 295
`\newcommand`, 75, 295
`\newcommand*`, 296
`\newcounter`, 125, 296
`\newenvironment`, 296
`\newenvironment*`, 296
`\newfont`, 195, 296
`\newlength`, 296
`\newline`, 16, 296
`\newpage`, 17, 296
`\newsavebox`, 95, 131, 296
`\newtheorem`, 151, 296
`\NG`, 297
`\ng`, 297
`\ni`, 62, 297
`\nobottombuttons`, 228

- \nobreakdash, 175
- \nocite, 297
- \nocorr, 26, 297
- \nofiles, 297
- \noindent, 21, 297
 - nointlimits, 154
- \nolimits, 47, 154, 297
- \nolinebreak, 17, 297
 - nonamelimits, 154
- \nonfrenchspacing, 297
- \nonumber, 57, 297
- \nopagebreak, 18, 297
- \normalcolor, 221, 297
- \normalfont, 25, 183, 297
- \normalmarginpar, 297
- \normalsize, 26, 297
 - nosumlimits, 154
- \not, 63, 297
- \notag, 153, 297
 - note, 230, 267
- \notesname, 297
- \notin, 63, 297
 - notitlepage, 67, 281
- \notopbuttons, 228
- \nu, 60, 298
- \numberwithin, 298
- \nwarrow, 63

- \O, 14, 298
- \o, 14, 298
- \oddsidemargin, 298
- \odot, 62, 298
- \OE, 14, 298
- \oe, 14, 298
- \oint, 64, 298
- \Omega, 60, 298
- \omega, 60, 298
- \ominus, 62, 298
- \onecolumn, 198, 298
 - onecolumn, 197, 281
 - oneside, 198, 281
- \onlynotes, 231, 298
- \onlyslides, 231, 298
 - openany, 199
 - openbib, 281
- \opening, 298
 - openright, 199
- \operatorname, 171
- \operatorname*, 171
- \oplus, 62, 298
- \OptionNotUsed, 298
 - origin, 222
- \oslash, 62, 298
- \otimes, 62, 299
- \oval, 89, 299
- \overbrace, 48, 299
- \overlay, 227
 - overlay, 230, 267
- \overlayempty, 227
- \overleftarrow, 174, 299
- \overleftrightharpoon, 174, 299
- \overline, 48, 299
- \overrightarrow, 174, 299
- \overset, 173, 299
 - oxtex, 107

- \P, 12, 299
- \PackageError, 299
- \PackageInfo, 299
- \PackageWarning, 299

-
- \PackageWarningNoLine, 299
 - page, 77
 - \pagebreak, 18, 299
 - \pagecolor, 221, 299
 - \pagedissolve, 228
 - \pagename, 300
 - \pagenumbering, 202, 300
 - \pageref, 161, 208, 300
 - \pagestyle, 200, 300
 - \paneloverlay, 227
 - \paneloverlayempty, 227
 - \paperheight, 300
 - \paperwidth, 300
 - \par, 16, 300
 - \paragraph, 68, 300
 - paragraph, 69, 77
 - \paragraph*, 300
 - \parallel, 62, 300
 - \parbox, 133, 300
 - \parindent, 21, 78, 300
 - \parsep, 126, 300
 - \parskip, 22, 78, 300
 - \part, 68, 301
 - part, 69, 77
 - \part*, 301
 - \partial, 64, 301
 - \partname, 301
 - \partopsep, 126, 301
 - \PassOptionsToClass, 301
 - \PassOptionsToPackage, 301
 - pc, 7
 - pctex32, 107
 - pctexhp, 107
 - pctexps, 107
 - pctexwin, 107
 - pcx, 104
 - \pdfannot, 224
 - \pdfdest, 225
 - \pdfendlink, 225
 - \pdfliteral, 223
 - pdfscreen, 226
 - \pdfstartlink, 225
 - pdftex, 107
 - \perp, 62, 301
 - \phantom, 20, 38
 - \Phi, 60, 301
 - \phi, 60, 301
 - \Pi, 60, 301
 - \pi, 60, 301
 - \Picture, 236
 - picture, 85, 267
 - \Picture*, 236
 - \Picture+, 236
 - plain, 200, 231
 - Plain TeX, 4
 - \pm, 62, 301
 - pmatrix, 163
 - \pmb, 154, 301
 - \pmod, 65, 301
 - \pod, 301
 - point (Tydraw), 99
 - polygon (Tydraw), 99
 - \poptabs, 134, 301
 - \popziti, 216
 - \pounds, 12, 301
 - \Pr, 301
 - \pr, 65
 - \Preamble, 235

- \prec, 62, 301
- \preceq, 62, 301
- \prefacename, 302
- \prime, 42, 64, 302
- \printindex, 209, 302
- \ProcessOptions, 302
- \ProcessOptions*, 302
- \prod, 64, 302
- \propto, 62, 302
- \protect, 302
- \providecommand, 302
- \providecommand*, 302
- \ProvidesClass, 302
- \ProvidesFile, 302
- \ProvidesPackage, 302
- \ProvideTextCommand, 192, 302
- \ProvideTextCommandDefault, 193, 303
- \ps, 303
- ps, 105
- \Psi, 60, 303
- \psi, 60, 303
- pt, 7
- \pushtabs, 134, 303
- \pushziti, 216
- \put, 86, 92, 303
- \qbezier, 93, 303
- \qquad, 19, 38, 172, 303
- \quad, 19, 38, 172, 303
- quotation, 120, 267
- quote, 120, 267
- \quotedblbase, 303
- \quotesinglbase, 303
- R, 229
- \r, 15, 303
- r, 33
- \raggedbottom, 199, 303
- \raggedleft, 120, 303
- \raggedright, 119, 303
- \raisebox, 132, 303
- \raisetag, 173, 303
- \rangle, 51, 303
- ratio (Tydraw), 100
- \rceil, 51, 303
- \Re, 64, 303
- \ref, 56, 161, 209, 303
- \reflectbox, 111, 304
- \refname, 29, 304
- \refstepcounter, 304
- \renewcommand, 75, 304
- \renewcommand*, 304
- \renewenvironment, 304
- \renewenvironment*, 304
- report, 66
- reqno, 154
- \RequirePackage, 304
- \RequirePackageWithOptions, 304
- \resizebox, 111, 304
- \resizebox*, 304
- \reversemarginpar, 304
- \rfloor, 51, 304
- \rhd, 62, 304
- \rho, 60, 305
- \right, 52, 305
- \right., 52
- \Rightarrow, 63, 305
- \rightarrow, 63, 305

- \rightharpoondown, 63, 305
\rightharpoonup, 63, 305
\rightleftharpoons, 63, 305
\rightmargin, 127, 305
\rm, 305
\rmdefault, 185, 305
\rmfamily, 24, 183, 305
\Roman, 124, 305
 Roman, 203
\roman, 124, 305
 roman, 203
\rotatebox, 112, 223, 305
\rq, 305
\rule, 132, 305
\rVert, 305
\rvert, 305

\S, 12, 305
\savebox, 95, 131, 305, 306
\sb, 306
\sbox, 95, 131, 306
\sc, 306
 scale, 222
\scalebox, 110, 306
\scdefault, 185, 306
\screensize, 227
\scriptscriptstyle, 37, 149, 189, 306
\scriptsize, 26, 306
\scriptstyle, 37, 149, 189, 306
\scshape, 25, 183, 306
\searrow, 63, 306
\sec, 65, 306
 secnumdepth, 69
\section, 68, 306
 section, 69, 77
\section*, 306
 section+, 235
\see, 306
\seename, 306
\selectfont, 182, 307
\selectlanguage, 307
\seriesdefault, 185
\setbmp (PCTeX32), 106
\setboolean, 307
\setcounter, 77, 307
\seteps (PCTeX32), 106
\setlength, 307
\SetMathAlphabet, 186, 307
\setminus, 62, 307
\setps (PCTeX32), 106
\SetSymbolFont, 307
\settime, 231, 307
\settodepth, 307
\settoheight, 307
\settowidth, 307
\setwmf (PCTeX32), 106
\sf, 307
\sfdefault, 185, 307
\sffamily, 24, 183, 307
 shadow (T_ydraw), 100
\shapedefault, 185
\sharp, 64, 308
\shortstack, 92, 308
\shoveleft, 156
\shoveright, 156
\showhyphens, 308
\sideset, 173, 308
\Sigma, 60, 308

- \sigma, 60, 308
- \signature, 308
- \sim, 62, 308
- \simeq, 62, 308
- \sin, 65, 308
- \sinh, 65, 308
- \sl, 308
- \sldefault, 185, 308
 - slide, 230, 267
 - slides, 229
- \sloppy, 308
 - sloppypar, 267
- \slshape, 25, 183, 308
- \small, 26, 308
 - smallmatrix, 164
- \smallskip, 21, 308
- \smallskipamount, 21, 78, 308
- \smile, 62, 308
- \songti, 216
- \sp, 308
- \spadesuit, 64, 308
- \special, 104, 106
 - Split, 229
 - split, 153, 156, 267
- \sqcap, 62, 308
- \sqcup, 62, 309
- \sqrt, 45, 309
- \sqsubset, 62, 309
- \sqsubseteq, 62, 309
- \sqsupset, 62, 309
- \sqsupseteq, 62, 309
- \srcsin, 65
- \SS, 14, 305
- \ss, 14, 309
- \stackrel, 49, 309
- \standardtilde, 212
- \star, 62, 309
- \stepcounter, 77, 309
- \stretch, 309
 - subarray, 165, 267
 - subequations, 267
- \subitem, 309
- \subjectname, 309
- \subparagraph, 68, 309
 - subparagraph, 69, 77
- \subparagraph*, 309
- \subsection, 68, 309
 - subsection, 69
- \subsection*, 309
- \subset, 62, 310
- \subseteq, 62, 310
- \substack, 164, 309
- \subsubitem, 309
- \subsubsection, 68, 309
 - subsubsection, 69, 77
- \subsubsection*, 310
- \succ, 62, 310
- \succeq, 62, 310
- \sum, 64, 310
 - sumlimits, 154
- \sup, 65, 310
- \suppressfloats, 310
- \supset, 62, 310
- \supseteq, 62, 310
- \surd, 64, 310
- \swarrow, 63, 310
- \symbol, 196, 310
- \t, 15, 310

- t, 33
- tabbing, 30, 267
- \tabbingsep, 135, 310
- \tabcolsep, 137, 310
- table, 114, 268
- table*, 114, 268
- \tablename, 310
- \tableofcontents, 204, 310
- tabular, 32, 135, 268
- tabular*, 136, 268
- \tabularnewline, 310
- \tag, 153, 310
- \tag*, 153
- \tan, 65, 311
- \tanh, 65, 311
- \tau, 60, 311
- \tbinom, 169, 311
- tbtags, 153
- tcidvi, 107
- \telephone, 311
- TeX, 3
- \TeX, 12, 311
- TeXdraw, 116
- \text, 155, 311
- textasciicircum, 311
- textasciitilde, 311
- textbackslash, 311
- textbar, 311
- \textbf, 25, 311
- textbullet, 311
- \textcircled, 311
- \textcolor, 221, 311
- \textcompwordmark, 311
- textemdash, 311
- textendash, 311
- textexclamdown, 311
- \textfloatsep, 311
- \textfraction, 311
- textgreater, 311
- \textheight, 78, 312
- \textit, 25, 312
- textless, 311
- \textmd, 25, 312
- \textnormal, 25, 312
- textperiodcentered, 311
- textquestiondown, 311
- \textquotedbl, 312
- textquotedblleft, 311
- textquotedblright, 311
- textquoteleft, 311
- textquoteright, 311
- \textregistered, 312
- \textrm, 25, 312
- \textsc, 25, 312
- \textsf, 25, 312
- \textsl, 25, 312
- \textstyle, 37, 149, 189, 312
- \textsuperscript, 312
- \texttrademark, 312
- \texttt, 25, 312
- \textup, 25, 312
- textures, 107
- textvisiblespace, 311
- \textwidth, 78, 312
- \text符号名, 311
- \tfrac, 168, 312
- \TH, 312
- \th, 312

- \thanks, 70, 312
- thebibliography, 29, 268
- \thefootnote, 140
- theindex, 268
- \Theta, 60, 313
- \theta, 60, 313
- \the计数器, 313
- \thicklines, 87, 313
- \thickspace, 172
- \thinlines, 87, 313
- \thinspace, 172, 313
- \thispagestyle, 202, 313
- threed (Tydraw), 101
- \Tilde, 166, 313
- \tilde, 48, 313
- \times, 62, 313
- \tiny, 26, 313
- \title, 70, 313
- titlepage, 67, 72, 268, 281
- \to, 63, 313
- \today, 313
- \top, 64, 313
- \topbuttons, 228
- \topfigrule, 313
- \topfraction, 313
- \topmargin, 313
- topnumber, 313
- \topsep, 126, 314
- \topskip, 314
- \totalheight, 129, 314
- totalheight, 222
- totalnumber, 314
- trace (Tydraw), 99
- \triangle, 64, 314
- \triangleleft, 62, 314
- \triangleright, 62, 314
- trim, 223
- trivlist, 268
- truetex, 107
- \tt, 314
- \ttdefault, 185, 314
- \ttfamily, 24, 183, 314
- \twocolumn, 198, 314
- twocolumn, 197, 281
- twod (Tydraw), 101
- twoside, 198, 281
- Tydraw, 5
- Tydraw, 98
- \typein, 207, 314
- \typeout, 207, 314
- \u, 15, 314
- \unboldmath, 60, 314
- \underbrace, 48, 315
- \underleftarrow, 174, 315
- \underleftrightharrow, 174, 315
- \underline, 48, 315
- \underrightarrow, 174, 315
- \underset, 173, 315
- \unitlength, 85, 315
- \unlhd, 62, 315
- \unrhd, 62, 315
- \Uparrow, 51, 63, 315
- \uparrow, 51, 63, 315
- \updefault, 185, 315
- \Updownarrow, 51, 63, 315
- \updownarrow, 51, 63, 315
- \uplus, 62, 315
- \uproot, 172

- \upshape, 25, 183, 315
\Upsilon, 60, 315
\upsilon, 60, 315
\urlid, 227
\usebox, 95, 131, 315
\usecounter, 125, 315
\usefont, 182, 316
\usepackage, 107, 316

\v, 15, 316
\value, 316
\varepsilon, 60, 316
\varinjlim, 316
\varliminf, 316
\varlimsup, 316
\varphi, 60, 316
\varpi, 60, 316
\varprojlim, 316
\varrho, 60, 316
\varsigma, 60, 316
\vartheta, 60, 316
\vdash, 62, 64, 316
\vdots, 38, 316
\Vec, 166
\vec, 48, 316
\vector, 87, 316
\vee, 62, 316
\verb, 121, 317
\verb*, 121, 317
 verbatim, 121, 268
 verbatim*, 121, 269
 verse, 120, 269
\vfill, 21, 317
 viewport, 223
\visible, 230, 317

\vline, 34, 317
 Vmatrix, 163, 269
 vmatrix, 163, 269
\voffset, 317
\vspace, 20, 317
\vspace*, 20

\wedge, 62, 317
\whiledo, 317
\widehat, 48, 317
\widetilde, 48, 317
\width, 129, 317
 width, 222
 width (T_ydraw), 99
 Wipe, 229
 wmf, 105
\wp, 64, 318
\wr, 62, 318
 write (T_ydraw), 100

 xdvi, 107
\Xi, 60, 318
\xi, 60, 318
\xleftarrow, 174, 318
\xrightarrow, 174, 318
 Xy-pic, 115

\zeta, 60, 318
\zihao, 216
\ziju, 216
\ziti, 216
\八, 24
\半, 24
\扁, 24
\标, 24
\大, 24

\仿, 23
\黑, 23
\九, 24
\巨, 24
\楷, 23
 控制符, 9
 控制字, 9
\阔, 24
\六, 24
\陆, 24
\七, 24
\叁, 24
\瘦, 24
\双, 24
\肆, 24
\宋, 23
\特, 24
\五, 24
\伍, 24
\小, 24
\正, 24
\中, 24

[G e n e r a l I n f o r m a t i o n]

书名 = L A T E X 入门与提高

作者 = 陈志杰

页数 = 3 5 0

S S 号 = 1 1 1 0 9 7 8 0

出版日期 = 2 0 0 2 年